

Fast Low-rank Approximation of a Matrix: Novel Insights, Novel Multipliers, and Extensions *

Victor Y. Pan^{[1,2],[a]}, Liang Zhao^{[2],[b]}, and John Svadlenka^{[2],[c]}

^[1] Department of Mathematics and Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA

^[2] Ph.D. Programs in Mathematics and Computer Science
The Graduate Center of the City University of New York
New York, NY 10036 USA

^[a] victor.pan@lehman.cuny.edu
<http://comet.lehman.cuny.edu/vpan/>

^[b] lzhaol@gc.cuny.edu

^[c] jsvadlenka@gradcenter.cuny.edu

Abstract

- Low-rank approximation of a matrix by means of random sampling has been consistently efficient in its empirical studies by many scientists who applied it with various sparse and structured multipliers, but adequate formal support for this empirical phenomenon has been missing so far.
- Our novel insight into the subject leads to such an elusive formal support and promises significant acceleration of the known algorithms for some fundamental problems of matrix computations and data mining and analysis.
- Our formal results and our numerical tests are in good accordance with each other.
- We also outline extensions of low-rank approximation algorithms and of our progress to the Least Squares Regression, the Fast Multipole Method, and the Conjugate Gradient algorithms.

Key Words: Low-rank approximation of a matrix, Random sampling, Derandomization, Least Squares Regression, Fast Multipole Method, Conjugate Gradient algorithms.

2000 Math. Subject Classification: 15A52, 68W20, 65F30, 65F20

1 Introduction

1.1 The problem of low-rank approximation and our progress briefly

Low-rank approximation of a matrix by means of random sampling is an increasingly popular subject area with applications to the most fundamental matrix computations [HMT11] as well as numerous problems of data mining and analysis, “ranging from term document data to DNA SNP data” [M11]. See [HMT11], [M11], and [GL13, Section 10.4.5], for surveys and ample bibliography.

All these studies rely on the proven efficiency of random sampling with Gaussian multipliers and mostly empirical evidence that the algorithms work as efficiently with various random sparse and structured multipliers, although adequate formal support for this empirical evidence has been missing so far.

*Some results of this paper have been presented at the Eleventh International Computer Science Symposium in Russia (CSR’2016), in St. Petersburg, Russia, June 2016

Our new insight enables such an elusive formal support as well as the acceleration of low-rank approximation and some other fundamental matrix computations. In this section we outline our main results by using the definitions below and in the Appendix.

1.2 Some definitions

- Typically we use the concepts “large”, “small”, “near”, “close”, “approximate”, “ill-conditioned” and “well-conditioned” quantified in the context, but we specify them quantitatively if needed.
- Hereafter “ \ll ” means “much less than” and “*flop*” stands for “floating point arithmetic operation”.
- I_s is the $s \times s$ identity matrix. $O_{k,l}$ is a $k \times l$ matrix filled with zeros. \mathbf{o} is a vector filled with zeros.
- $(B_1 \mid \dots \mid B_h)$ denotes a $1 \times h$ block matrix with the blocks B_1, \dots, B_h .
- $\text{diag}(B_1, \dots, B_h)$ denotes a $h \times h$ block diagonal matrix with diagonal blocks B_1, \dots, B_h .
- $\text{rank}(M)$, $\text{nrank}(M)$, and $\|M\|$ denote the *rank*, *numerical rank*, and the *spectral norm* of a matrix M , respectively. $[\text{nrank}(M) = r]$ if and only if a matrix $M - E$ is well-conditioned and has rank r , for a perturbation matrix E of a small norm. A matrix is ill-conditioned if and only if its rank exceeds its numerical rank or equivalently if and only if its small-norm perturbation can decrease its rank.]
- M^T , M^H , and $\mathcal{R}(M)$ denote its transpose, Hermitian transpose, and range (column span), respectively.
- An $m \times n$ matrix M is called *unitary* if $M^H M = I_n$ or if $M M^H = I_m$. If this matrix is known to be real, then it is also and preferably called *orthogonal*.
- “*Likely*” means “with a probability close to 1”, the acronym “i.i.d.” stands for “independent identically distributed”, and we refer to standard Gaussian random variables just as *Gaussian*.
- We call an $m \times n$ matrix *Gaussian* and denote it $G_{m,n}$ if all its entries are *i.i.d.* Gaussian variables.
- $\mathcal{G}^{m \times n}$, $\mathbb{R}^{m \times n}$, and $\mathbb{C}^{m \times n}$ denote the classes of $m \times n$ Gaussian, real and complex matrices, respectively.
- $\mathcal{G}_{m,n,r}$, $\mathbb{R}_{m,n,r}$, and $\mathbb{C}_{m,n,r}$, for $1 \leq r \leq \min\{m, n\}$, denote the classes of $m \times n$ matrices UV of rank at most r where the matrices U of size $m \times r$ and V of size $r \times n$ are Gaussian, real and complex, respectively.
- If $M = UV \in \mathcal{G}_{m,n,r}$, then the matrices U , V and M have rank r with probability 1 by virtue of Theorem B.1, and such a matrix M is said to be an $m \times n$ factor-Gaussian matrix of expected rank r .

1.3 The basic algorithm

Recall that a matrix M can be represented (respectively, approximated) by a product UV of two matrices $U \in \mathbb{C}^{m \times r}$ and $V \in \mathbb{C}^{r \times n}$ if and only if $r \geq \text{rank}(M)$ (respectively, $r \geq \text{nrank}(M)$). The following celebrated algorithm computes such a representation or approximation:

Algorithm 1.1. [HMT11, Algorithm 4.1]. Low-rank representation/approximation of a matrix.

INPUT: An $m \times n$ matrix M , a nonnegative tolerance τ , and an integer r such that $0 < r \ll \min\{m, n\}$.

INITIALIZATION: Fix an integer p such that $0 \leq p \ll n - r$. Compute $l = r + p$. Generate an $n \times l$ matrix B .

COMPUTATIONS: 1. Compute the $m \times l$ matrix MB .

2. By orthogonalizing its columns, compute and output the $m \times l$ matrix $Q = Q(MB)$ (cf. [GL13, Theorem 5.2.3]).

3. Compute and output the $l \times n$ matrix $Q^T M$ (together with the matrix Q it represents the $m \times n$ matrix $\tilde{M} = Q Q^T M$).

4. Compute and output the spectral error norm $\Delta = \|\tilde{M} - M\|$.¹ If $\Delta \leq \tau$, output *SUCCESS*. Otherwise output *FAILURE*.

Hereafter we focus on Stage 1, involving $(2n - 1)ml$ flops for generic matrices M and B . The range of its output matrix MB approximates the left leading singular space \mathcal{S}_r of the input matrix M associated with its r largest singular values, and [HMT11] call this stage “Range finder”. Within the same asymptotic cost bound, [HMT11, Section 5] extends it to the approximation of the SVD of the matrix M , producing its low-rank approximation, and to some other fundamental factorizations of M .

It remains to specify the choice of a multiplier B in order to complete the description of the algorithm.

1.4 The choice of multipliers: basic observations

We readily prove the following result (see Section 4.3):

Theorem 1.1. *Given an $m \times n$ matrix M with $\text{nrnk}(M) = r$ and a reasonably small positive tolerance τ , Algorithm 1.1 outputs *SUCCESS* if and only if $\text{nrnk}(MB) = r$.*

Hence, for a given matrix M and a proper choice of an $n \times l$ multiplier B , the algorithm solves the *fixed rank problem* where the integer $r = \text{nrnk}(M)$ is given (we could have estimated this integer by means of binary search based on recursive application of the algorithm).

Definition 1.1. *For two integers l and n , $0 < l \leq n$, and any fixed $n \times l$ multiplier B , partition the set of $m \times n$ matrices M with $\text{nrnk}(M) = r$ into the sets $\mathcal{M}_B = \mathcal{M}_{B,\text{good}}$ and $\mathcal{M}_{B,\text{bad}}$ of “ B -good” and “ B -bad” matrices such that $\text{nrnk}(MB) = r$ and $\text{nrnk}(MB) < r$, respectively.*

The following simple observations should be instructive.

Theorem 1.2. (Cf. Remark 1.1.) *Consider a vector \mathbf{v} of dimension n and unitary matrices Q of size $n \times n$, Q' of size $l \times l$, and B of size $n \times l$, so that both matrices QB and BQ' have size $l \times l$ and are unitary. Then*

- (i) $\mathcal{M}_{B'} = \mathcal{M}_B$,
- (ii) $\mathcal{M}_{B''} = (\mathcal{M}_B)Q$,
- (iii) $\mathcal{M}_B \subseteq \mathcal{M}_{(B \mid \mathbf{v})}$, and
- (iv) \mathcal{M}_B is the set of all $m \times n$ matrices having numerical rank r (that is, the set $\mathcal{M}_{B,\text{bad}}$ of B -bad matrices is empty) if and only if $l \geq n$.

The theorem implies that the class \mathcal{M}_B of B -good matrices is invariant in the map $B \rightarrow BQ'$ (by virtue of part (i)), is invariant (up to its orthogonal transformation) in the map $B \rightarrow QB$ (by virtue of part (ii)), can only grow if a column vector is appended to a multiplier B (by virtue of part (iii)), and fills the whole space $\mathbb{C}_{m,n,r}$ or $\mathbb{R}_{m,n,r}$ if $l = n$ (by virtue of part (iv)).

1.5 A recursive algorithm

Theorem 1.2 motivates the design of the following algorithm.

Algorithm 1.2. Recursive low-rank representation/approximation of a matrix.

INPUT: An $m \times n$ matrix M and a nonnegative tolerance τ .

COMPUTATIONS: 1. Generate an $n \times n$ unitary matrix B .

2. Fix positive integers l_1, \dots, l_h such that $l_1 + \dots + l_h = n$ and represent the matrix B as a block vector $B = (B_1 \mid \dots \mid B_h)$ where the block B_i has size $n \times l_i$ for $i = 1, \dots, h$.
3. Recursively apply Algorithm 1.1 to the matrix M by substituting l for $l^{(i)} = \sum_{j=1}^i l_j$ and B for $B^{(i)} = (B_1 \mid \dots \mid B_i)$, $i = 1, \dots, h$. Stop when the algorithm outputs *SUCCESS*.

Correctness of the algorithm follows from part (iv) of Theorem 1.2.

¹Frievalds’ test of [F77] enables probabilistically estimation of the norm Δ at a cost of performing a bounded number of multiplications of the matrix $\tilde{M} - M$ by random vectors.

Remark 1.1. We can extend both Theorem 1.2 and Algorithm 1.2 to the case where we apply them to a nonsingular and well-conditioned (rather than unitary) $n \times n$ matrix B . In that case all blocks B_j of the multipliers $B^{(i)}$ are also well-conditioned matrices of full rank, and moreover $\kappa(B_j) \leq \kappa(B)$ for all j (cf. [GL13, Corollary 8.6.3]).

At every recursive stage of the algorithm we can reuse the results of the previous stages (see Section 2). If the algorithm outputs SUCCESS at Stage $l^{(h)}$, then its overall computational cost is bounded by that of Algorithm 1.1 for the same input and for $l = l^{(h)}$. The bound decreases as the output dimension $l^{(h)}$ decreases, which motivates our next goal of the *compression of the output multipliers*, that is, of yielding success for a smaller dimension $l^{(h)}$.

1.6 Compression of output multipliers by means of Gaussian and structured randomization

The following theorem implies that Algorithm 1.2 is likely to output SUCCESS at Stage h for the smallest h such that $l^{(h)} \geq r$ if B is a Gaussian matrix.

Theorem 1.3. Let Algorithm 1.1 be applied with a Gaussian multiplier $B = G_{n,l}$. Then

- (i) $\tilde{M} = M$ with probability 1 if $l = r = \text{rank}(M)$ (cf. Theorem 4.1) and
- (ii) it is likely that $\tilde{M} \approx M$ if $\text{nrnk}(M) = r \leq l - 4$ (cf. Theorem 4.3).

We produce an $n \times l$ Gaussian matrix $B = G_{n,l}$ by generating its nl random entries, and we pre-multiply it by a dense $m \times n$ matrix M by using $ml(2n - 1)$ flops. (They dominate $O(ml^2)$ flops for the orthogonalization at Stage 2 if $l \ll n$.) We produce an $n \times l$ matrix B of *subsample random Fourier or Hadamard transform*² by generating $n + l$ parameters, which define this matrix, and we pre-multiply it by an $m \times n$ matrix M by using $O(mn \log(l))$ flops, for l of order $r \log(r)$ (see [HMT11, Sections 4.6 and 11], [M11, Section 3.1], and [T11]).

SRFT and SRHT multipliers B are *universal*, like Gaussian ones: Algorithm 1.1 applied with such a multiplier is likely to approximate closely a matrix M having numerical rank at most r , although the transition from Gaussian to SRFT and SRHT multipliers increases the estimated failure probability from $3\exp(-p)$, for $p \geq 4$, to $O(1/l)$ (cf. [HMT11, Theorems 10.9 and 11.1], [M11, Section 5.3.2], and [T11]).

Empirically Algorithm 1.1 with SRFT multipliers fails very rarely even for $l = r + 20$, although for some special input matrices M it is likely to fail if $l = o(r \log(r))$ (cf. [HMT11, Remark 11.2] or [M11, Section 5.3.2]). Researchers have consistently observed similar empirical behavior of the algorithm applied with SRHT and various other multipliers (see [HMT11], [M11], [W14], [PQY15], and the references therein),³ but so far no adequate formal support for that empirical observation has appeared in the huge bibliography on this highly popular subject.

1.7 Our goals, our dual theorem, and its implications

In this paper we are going to

- (i) fill the void in the bibliography by supplying a missing *formal support* for the cited observation, with far reaching implications (see parts (ii) and (iv) below),
- (ii) define *new more efficient policies of generation and application of multipliers* for low-rank approximation,
- (iii) *test our policies numerically*, and
- (iv) *extend our progress* to other important areas of matrix computations.

Our basic *dual* Theorem 1.4 below reverses the assumptions of our basic *primal* Theorem 1.3 that a multiplier B is Gaussian, while a matrix M is fixed.

²Hereafter we use the acronyms *SRFT* and *SRHT*.

³In view of part (ii) of Theorem 1.2, the results cited for the classes of SRFT and SRHT matrices also hold for the products of these classes with any unitary matrix, in particular for the class of random $n \times l$ blocks of $n \times n$ circulant matrices (see their definitions, e.g., in [P01]).

Theorem 1.4. Let $M - E \in \mathcal{G}_{m,n,r}$ and $\|E\|_2 \approx 0$ (so that $\text{nrank}(M) \leq r$ and that $\text{nrank}(M) = r$) with probability 1. Furthermore let $B \in \mathbb{R}^{n \times l}$ and $\text{nrank}(B) = l$. Then

- (i) Algorithm 1.1 outputs a rank- r representation of a matrix M with probability 1 if $E = 0$ and if $l = r$ and
- (ii) it outputs a rank- l approximation of that matrix with a probability close to 1 if $p = l - r > 3$ and approaching 1 fast as the integer p grows from 4.

Theorem 1.4 implies that Algorithm 1.1 outputs a low-rank approximation to the average input matrix M that has a small numerical rank r (and thus in a sense to most of such matrices) if the multiplier B is a well-conditioned matrix of full rank and if the average matrix is defined under the Gaussian probability distribution. The former provision, that $\text{nrank}(B) = l$, is natural for otherwise we could have replaced the multiplier B by an $n \times l_-$ matrix for $l_- < l$. The latter customary provision is natural in view of the Central Limit theorem.

As an immediate implication of the theorem, on the average input M Algorithm 1.2 applied with any unitary or just nonsingular and well-conditioned $n \times n$ multiplier B also outputs SUCCESS at its earliest recursive Stage h at which the dimension $l^{(h)} = \sum_{j=1}^h l_j$ exceeds $r - 1$. Such application can be viewed as *derandomization* of Algorithms 1.1 and 1.2 versus their common application with Gaussian sampling.

In Sections 2 and 3 we specify some promising policies of the generation of multipliers based on these observations and in Section 5 perform numerical tests for such policies.

1.8 Related work, our novelties, and extension of our progress

Part (ii) of our Theorem 1.3 is implied by [HMT11, Theorem 10.8], but our specific supporting estimates are more compact, cover the case of any $l \geq r$ (whereas [HMT11] assumes that $l \geq r + 4$), and we deduce them by using a shorter proof (see Remark 4.4). Our approach, our results listed in Section 1.7, some of our techniques, e.g., expansion/compression in Section 2, and even the concept of factor-Gaussian matrices are new, and so are our families of multipliers and policies of their generation, combination, and application in Sections 2 and 3 as well.

Moreover, our progress can be extended to a variety of important matrix computations. In Section 6 (Conclusions) we outline such novel extensions to the highly popular and much studied computations for Least Squares Regression, the Fast Multipole Method and the Conjugate Gradient Algorithms.⁴ The extensions provide new insights and new opportunities and should motivate further effort and further progress.

1.9 Organization of the paper

We organize our presentation as follows:

- In Section 2 we describe some policies for managing rare failures of Algorithm 1.1.
- In Section 3 we present some efficient multipliers for low-rank approximation.
- In Section 4 we prove Theorems 1.1, 1.3, and 1.4.
- Section 5 (the contribution of the second and the third authors) covers our numerical tests.
- In Section 6 we extend our approach to the acceleration of the LSR, FMM, and CG algorithms.
- The Appendix covers basic definitions, auxiliary results, and low-rank representation of a matrix.

2 Preventing and managing unlikely failure of Algorithm 1.1

Two conflicting goals and a simple recipe.

We try to decrease:

- (i) the cost of generation of a multiplier B and of the computation of the matrices MB , $Q = Q(MB)$, $Q^T M$, and $QQ^T M - M$ and

⁴Hereafter we use the acronyms “LSR” for “Least Squares Regression”, “FMM” for “Fast Multipole Method”, and “CG” for “Conjugate Gradient”.

(ii) the chances for failure of Algorithm 1.1.

Towards these goals we should seek sparse and structured multipliers B_1, \dots, B_h such that Algorithm 1.2 would succeed already for $h = 1$. Theorem 1.4 implies that this holds for the average $m \times n$ matrix M having a small numerical rank r whenever we apply an $n \times l$ well-conditioned multiplier B of full rank $l \geq r$. Thus we readily achieve our goals except for relatively rare special matrices M .

Real computations can deal with “rare special” input matrices M , not covered by Theorem 1.4, but in our extensive tests in Section 5 with a variety of inputs, our small collection of sparse and structured multipliers in the next section turned out to be powerful enough for fulfilling our goals.

In the rest of this section we discuss the choice of the size of the multipliers and the policies of handling the cases where Algorithm 1.2 needs more than a single recursive step in order to succeed.

The size of multipliers. In the case of generic input matrix M , Algorithm 1.1 has the same chances for success with any well-conditioned multiplier B of full rank l , by virtue of Theorem 1.2. Empirically these chances are quite high and increase fast as the integer parameter $p = l - r$ grows, but then the cost of computing the matrices MB , $Q = Q(MB)$, $Q^T M$, and $QQ^T M - M$ increases as well, contrary to our goal (i). So heuristic choice of a positive integer p below 20 is preferred: “setting $l = r + 10$ or $l = r + 20$ is typically more than adequate” according to [HMT11, Section 4.6] and to our extensive tests.

Reusing multipliers. Recall from [HMT11, Sections 8 and 9] that for all j the matrices $\tilde{M}^{(j)} = Q^{(j)}Q^{(j)T}M$ and $\tilde{M}_j = Q_jQ_j^T M$, for $Q^{(j)} = Q(MB^{(j)})$, $Q_j = Q(MB_j)$, and $B^{(j)} = (B_1 \mid \dots \mid B_j)$, are the orthogonal projections of the matrices $MB^{(j)}$ and MB_j , respectively, onto the range of the matrix M . Hence $M - \tilde{M}^{(h)} = M - \tilde{M}^{(h-1)} - \tilde{M}_h$, and so at the h th stage of Algorithm 1.2, for $h > 1$, we can reuse such projections computed at its stage $h - 1$ rather than recompute them.

Compression of low-rank approximations via expansion and with heuristic amendment.

Consider the following **expansion/compression algorithm**:

1. Fix a larger dimension l_+ , generate a sparse and structured $n \times l_+$ multiplier B , and compute the $m \times l_+$ product MB .
2. Fix an integer l , $r \leq l \leq fl_+$ for a reasonably small fraction f , generate a Gaussian $l_+ \times l$ multiplier G and compress the product MB into the $m \times l$ matrix MBG .

For appropriate sparse and structured multipliers B , Stage 1 is inexpensive even for $l_+ = n$. Moreover, by virtue of Theorem 1.4, the class of input matrices M for which $(MB) \approx \mathcal{S}_r$ is large and grows fast as the integer $l_+ - r$ increases from 4. Stage 2 produces a matrix MBG , such that it is likely that $(MBG) \approx \mathcal{S}_r$ by virtue of Theorem 1.3 and involves $(2l_+ - 1)ml$ flops, which is relatively inexpensive if $l - r$ is not large and if $l_+ \ll \min\{m, n\}$. Furthermore we can decrease this cost by applying a SRHT or STRF multiplier G instead of Gaussian (cf. [HMT11, Section 4.6]).

In our extensive tests in Section 5, even a simpler heuristic solution was always sufficient for the compression of the output. Namely, in our tests, Algorithm 1.2 consistently succeeded after trying $h < 4$ multipliers properly chosen (in line with Theorem 1.4), and for $h > 1$ we produced a desired compressed low-rank approximation by applying the following heuristic amendment to the algorithm:

Amendment 1 (combining failed multipliers): if Algorithm 1.2 has succeeded in $h > 1$ recursive steps, then re-apply it with the sum or linear combination of the h multipliers B_1, \dots, B_h involved into these steps.

3 Generation of Multipliers

3.1 Multipliers as rectangular submatrices

Given two integers l and n , $l \ll n$, we first generate four classes of very sparse primitive $n \times n$ matrices, then combine them into some basic families of $n \times n$ matrices (we denote them \hat{B} in this section), and finally define efficient $n \times l$ multipliers as submatrices B , made up of l fixed (e.g., leftmost) or random columns of the matrices \hat{B} . In this case $\kappa(B) \leq \kappa(\hat{B})$ (cf. [GL13, Theorem 8.6.3]).

Random choice of l columns involves l random parameters, but in the transition $\hat{B} \rightarrow B$ we lose $n - l$ columns of the matrix \hat{B} together with all random parameters (if any) located in these columns. The

arithmetic cost of the computation of the product MB does not increase in the transition $\widehat{B} \rightarrow B$, but can stay invariant, e.g., if the matrix \widehat{B} is structured and if the transition is by means of random choice of l columns of the matrix \widehat{B} .

The transition $\widehat{B} \rightarrow B$ to a $p \times q$ matrix B where $\max\{p, q\} < n$ can be numerically unstable: it can increase the condition number dramatically. This observation restricts the proposed basic families (i) and (iii) of $n \times n$ multipliers of Sections 3.4 and 3.6 to the case where n is a power of 2. By using block representation of a matrix \widehat{B} , however, we can relax this restriction in the cases where $n = \sum_{i=1}^q (-1)^{m_i} 2^{k_i}$ is an algebraic sum of a small number q of powers of 2: in such cases we can use matrices of basic families (i) and (iii) as blocks of sizes $2^u \times 2^v$ for integers u and v . We refer the reader to [M11], [W14], and the bibliography therein for various other techniques for relaxing the restrictions on the size of multipliers.

3.2 $n \times n$ matrices of four primitive types

1. Fixed or random *permutation matrix* P .
2. A *diagonal matrix* $D = \text{diag}(d_i)_{i=0}^{n-1}$, with fixed or random diagonal entries d_i such that $|d_i| = 1$ for all i (and so each of n entries d_i lies on the unit circle $\{z : |z| = 1\}$, being either nonreal or ± 1).
3. An f -*circular shift matrix* $Z_f = \begin{pmatrix} 0^T & f \\ I_{n-1} & 0 \end{pmatrix}$ and its transpose Z_f^T for a fixed scalar f such that either $f = 0$ or $|f| = 1$. We write $Z = Z_0$, call Z *unit down-shift matrix*, and call Z_1 *unit circulant matrix*.
4. A $2s \times 2s$ *Hadamard primitive matrix* $H^{(2s)} = \begin{pmatrix} I_s & I_s \\ I_s & -I_s \end{pmatrix}$ for a positive integer s (cf. [M11], [W14]).

All these multipliers are very sparse, have nonzero entries evenly distributed throughout them, and can be pre-multiplied by a vector by using from 0 to $2n$ flops per a multiplier. All of them, except for the matrix Z , are unitary (or real orthogonal). Hence, for the average input matrix M , Algorithm 1.1 succeeds with any of their $n \times l$ submatrix B by virtue of Theorem 1.4, and similarly with any $n \times l$ submatrix of Z of full rank. For sparse matrices M , the algorithm can fail with these sparse multipliers in good accordance with our tests (cf. Example 3.1 below), but in all these tests we have consistently succeeded with fixed (e.g., leading) or random $n \times l$ submatrices of various simple *combinations* of our four primitives.

Example 3.1. Let B denote an $n \times r$ matrix. Then $\text{rank}(MB) = r$ for $M = \text{diag}(I_r, O_{m-r, n-r})P$ and all permutation matrices P if and only if all $r \times r$ submatrices of the multiplier B are nonsingular.

3.3 Basic combinations of primitive matrices: general observations

In the next subsections, by combining primitives 1–4, we define families of $n \times n$ sparse and/or structured matrices whose $n \times l$ submatrices B have been successfully tested as multipliers in Section 5.

For the choice of permutation and diagonal primitive matrices trade-off is possible: by choosing the identity matrix I_n for these primitives we decrease the computational cost of the generation and application of the multipliers, whereas by choosing random primitives we increase this cost but also increase the chances for success of Algorithm 1.1.

3.4 Family (i): multipliers based on the Hadamard and Fourier processes

At first we recursively define the dense and orthogonal (up to scaling by constants) $n \times n$ matrices H_n of *Walsh-Hadamard transform* for $n = 2^k$ as follows (cf. [M11, Section 3.1] and our Remark 3.1):

$$H_{2q} = \begin{pmatrix} H_q & H_q \\ H_q & -H_q \end{pmatrix} \quad (3.1)$$

for $q = 2^h$, $h = 0, 1, \dots, k-1$, and the Hadamard primitive matrix $H_2 = H^{(2)} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ of type 4 for $s = 1$.

We can pre-multiply such a matrix H_n by a vector by using nk additions and subtractions for $n = 2^k$.

Next we sparsify it by defining it by a shorter recursive process, that is, by fixing a *recursion depth* d , $0 < d < k$, and applying equation (3.1) where $q = 2^h$, $h = k-d, k-d+1, \dots, k-1$, and H_{2s} for $2s = 2^{k-d+1}$

is the Hadamard primitive matrix $H^{(2s)}$ of type 4. For $n = 2^k$, we denote the resulting $n \times n$ matrix $H_{n,d}$ and for $1 \leq d < k$ call it *d-Abridged Hadamard (AH) matrix*.

This matrix is still orthogonal (up to scaling), has $q = 2^d$ nonzero entries in every row and column, and hence is sparse unless $k - d$ is a small integer. Pre-multiplication of such a matrix by a vector uses dn additions and subtractions and allows highly efficient parallel implementation (cf. Remark 3.2).

We similarly obtain sparse matrices by shortening a recursive process of the generation of the $n \times n$ matrix Ω_n of *discrete Fourier transform (DFT)* at n points, for $n = 2^k$:

$$\Omega_n = (\omega_n^{ij})_{i,j=0}^{n-1}, \text{ for } n = 2^k \text{ and a primitive } n\text{th root of unity } \omega_n = \exp(2\pi\sqrt{-1}/n). \quad (3.2)$$

The matrix Ω_n is unitary up to scaling by $\frac{1}{\sqrt{n}}$. We can multiply it by a vector by using $1.5nk$ flops (by applying FFT) and can generate it recursively as follows (cf. [P01, Section 2.3] and our Remark 3.1):⁵

$$\Omega_{2q} = \hat{P}_{2q} \begin{pmatrix} \Omega_q & \Omega_q \\ \Omega_q \hat{D}_q & -\Omega_q \hat{D}_q \end{pmatrix}, \quad (3.3)$$

for $q = 2^h$, $h = 0, 1, \dots, k$, and $\Omega_2 = H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

Then again we can sparsify this matrix by defining it by a shorter recursive process, that is, by fixing a recursion depth d , $0 < d < k$, and applying equation (3.3) for $q = 2^h$, $h = k - d, k - d + 1, \dots, k - 1$, and Ω_{2s} for $2s = 2^{k-d+1}$ being the Hadamard primitive matrix $H^{(2s)}$ of type 4.

For $1 \leq d \leq k$ and $n = 2^k$, we denote the resulting $n \times n$ matrix $\Omega_{n,d}$ and call it *d-Abridged Fourier (AF) matrix*. It is also unitary (up to scaling), has $q = 2^d$ nonzero entries in every row and column, and thus is sparse unless $k - d$ is a small integer. Pre-multiplication of such a matrix by a vector involves $1.5dn$ flops and again allows highly efficient parallel implementation (cf. Remark 3.2).

By applying fixed or random permutation and/or scaling to AH matrices H_n and AF matrices Ω_n , we obtain the sub-families of *d-Abridged Scaled and Permuted Hadamard (ASPH)* matrices, PDH_n , and *d-Abridged Scaled and Permuted Fourier (ASPF)* $n \times n$ matrices, $PD\Omega_n$. Then we define the families of ASH, ASF, APH, and APF matrices, DH_n , $D\Omega_n$, PH_n , and $P\Omega_n$, respectively. Each random permutation or scaling contributes up to n random parameters.

Remark 3.1. We can readily express representations (3.1) and (3.3) as combinations of our primitives 1–4:

$$H_{2q} = \text{diag}(H_q, H_q)H^{(2q)} \text{ and } \Omega_{2q} = \hat{P}_{2q} \text{diag}(\Omega_q, \Omega_q \hat{D}_q)H^{(2q)}$$

where $H^{(2q)}$ denotes a $2q \times 2q$ Hadamard's primitive matrix of type 4. We can introduce more random parameters by means of random recursive permutations and diagonal scaling as follows:

$$\hat{H}_{2q} = P_{2q}D_{2q} \text{diag}(\hat{H}_q, \hat{H}_q)H^{(2q)} \text{ and } \hat{\Omega}_{2q} = P_{2q}D_{2q} \text{diag}(\Omega_q, \Omega_q \hat{D}_q)H^{(2q)}$$

where P_{2q} are $2q \times 2q$ random permutation matrices of primitive class 1 and D_{2q} are $2q \times 2q$ random matrices of diagonal scaling of primitive class 2, for all q . We need at most $2dn$ additions and subtractions in order to pre-multiply a matrix \hat{H}_n by a vector and at most $2.5dn$ flops in order to pre-multiply a matrix \hat{H}_n by a vector, in both cases for n being a power of 2.

3.5 f -circulant and sparse f -circulant matrices

Recall that an f -circulant matrix $Z_f(\mathbf{v}) = \sum_{i=0}^{n-1} v_i Z_f^i$, for the matrix Z_f of f -circular shift, is defined by a scalar $f \neq 0$ and by the first column $\mathbf{v} = (v_i)_{i=0}^{n-1}$ and is called *circulant* if $f = 1$ and *skew-circulant* if $f = -1$. Such a matrix is nonsingular with probability 1 (see Theorem B.1) and is likely to be well-conditioned [PSZ15] if $|f| = 1$ and if the vector \mathbf{v} is Gaussian or is random and uniformly bounded.

FAMILY (ii) of *sparse f -circulant matrices* $\hat{B} = Z_f(\mathbf{v})$ is defined by a fixed or random scalar f , $|f| = 1$, and by the first column having exactly q nonzero entries, for $q \ll n$. The positions and values of nonzeros can be randomized (and then the matrix would depend on up to $2n + 1$ random values).

⁵Transposition of the matrices of this representation of FFT, called decimation in frequency (DIF) radix-2 representation, turns them into the matrices of the alternative representation of FFT, called decimation in time (DIT) radix-2 representation.

Such a matrix can be pre-multiplied by a vector by using at most $(2q - 1)n$ flops or, in the real case where $f = \pm 1$ and $v_i = \pm 1$ for all i , by using at most qn additions and subtractions.

The same cost estimates apply to the generalization of such a matrix $Z_f(\mathbf{v})$ to any sparse matrix with exactly q nonzero entries ± 1 in every row and in every column for $1 \leq q \ll n$, which can be defined as the sum $\sum_{i=1}^q \hat{D}_i P_i$ for fixed or random matrices P_i and \hat{D}_i of primitive types 1 and 2, respectively.

3.6 Abridged f -circulant matrices

First recall the following well-known expression for a g -circulant matrix:

$$Z_g(\mathbf{v}) = \sum_{i=0}^{n-1} v_i Z_g^i = D_f^{-1} \Omega_n^H D \Omega_n D_f$$

where $g = f^n$, $D_f = \text{diag}(f^i)_{i=0}^{n-1}$, $\mathbf{v} = (v_i)_{i=0}^{n-1} = (\Omega_n D_f)^{-1} \mathbf{d}$, $\mathbf{d} = (d_i)_{i=0}^{n-1}$, and $D = \text{diag}(d_i)_{i=0}^{n-1}$ (cf. [P01, Theorem 2.6.4]). For $f = 1$, the expression is simplified: $g = 1$, $D_f = I_n$, $F = \Omega_n$, and $Z_g(\mathbf{v}) = \sum_{i=0}^{n-1} v_i Z_1^i$ is a circulant matrix:

$$Z_1(\mathbf{v}) = \Omega_n^H D \Omega_n, \quad D = \text{diag}(d_i)_{i=0}^{n-1}, \quad \text{for } \mathbf{d} = (d_i)_{i=0}^{n-1} = \Omega_n \mathbf{v}.$$

Pre-multiplication of an f -circulant matrix by a vector is reduced to pre-multiplication of the matrices Ω and Ω^H by two vectors and in addition to $4n$ flops (or $2n$ in case of a circulant matrix). This involves $O(n \log(n))$ flops overall and then again allows highly efficient parallel implementation (see Remark 3.2).

For a fixed scalar f , we can define the matrix $C_g(\mathbf{v})$ by any of the vectors \mathbf{v} or $\mathbf{d} = (d_i)_{i=0}^{n-1}$. The matrix is unitary (up to scaling) if $|f| = 1$ and if $|d_i| = 1$ for all i . The family of such matrices is defined by $n + 1$ real parameters (or by n such parameters for a fixed f) which we can fix or choose at random.

Now suppose that $n = 2^k$, $1 \leq d < k$, d and k are integers, and substitute a pair of AF matrices of recursion length d for two factors Ω_n in the above expressions. Then the resulting *abridged f -circulant matrix* $C_{g,d}(\mathbf{v})$ of *recursion depth* d is still unitary (up to scaling), defined by $n + 1$ or n parameters d_i and f , is sparse unless the positive integer $k - d$ is small, and can be pre-multiplied by a vector by using $(3d + 3)n$ flops. Instead of AF matrices, we can substitute a pair of ASPF, APF, ASF, AH, ASPH, APH, or ASF matrices for the factors Ω_n . Such matrices form **FAMILY (iii)** of *d -abridged f -circulant matrices*.

3.7 Inverses of bidiagonal matrices

FAMILY (iv) is formed by the *inverses of bidiagonal matrices*

$$\hat{B} = (I_n + DZ)^{-1} \text{ or } (I_n + Z^T D)^{-1}$$

for a matrix D of primitive type 2 and the down-shift matrix Z .

We can randomize the matrix \hat{B} by choosing up to $n - 1$ random diagonal entries of the matrix D (whose leading entry makes no impact on \hat{B}) and can pre-multiply the matrix \hat{B} by a vector by using $2n - 1$ flops or, in the real case, just $n - 1$ additions and subtractions.

$\|\hat{B}\| \leq \sqrt{n}$ because nonzero entries of the lower triangular matrix $\hat{B} = (I_n + DZ)^{-1}$ have absolute values 1, and clearly $\|\hat{B}^{-1}\| = \|I_n + DZ\| \leq \sqrt{2}$. Hence $\kappa(\hat{B}) = \|\hat{B}\| \|\hat{B}^{-1}\|$ (the spectral condition number of \hat{B}) cannot exceed $\sqrt{2n}$ for $\hat{B} = (I_n + DZ)^{-1}$, and the same bound holds for $\hat{B} = (I_n + Z^T D)^{-1}$.

3.8 Summary of estimated numbers of flops and random variables involved

Table 3.1 shows upper bounds on (a) the numbers of random variables involved into the matrices \hat{B} of the four families (i)–(iv) and (b) the numbers of flops for pre-multiplication of such a matrix by a vector.⁶ For comparison, using a Gaussian $n \times n$ multiplier involves n^2 random variables and $(2n - 1)n$ flops.

⁶The asterisks in the table show that the matrices of families (i) AF, (i) ASPF, and (iii) involve nonreal roots of unity.

Table 3.1: The numbers of random variables and flops

family	(i) AH	(i) ASPH	(i) AF	(i) ASPF	(ii)	(iii)	(iv)
random variables	0	$2n$	0	$2n$	$2q + 1$	n	$n - 1$
flops complex	dn	$(d + 1)n$	$1.5dn$	$(1.5d + 1)n$	$(2q - 1)n$	$(3d + 2)n$	$2n - 1$
flops in real case	dn	$(d + 1)n$	*	*	qn	*	$n - 1$

Remark 3.2. Other observations besides flop estimates can be decisive. E. g., a special recursive structure of an ARSPH matrix $H_{2^k, d}$ and an ARSPF matrix $\Omega_{2^k, d}$ allows highly efficient parallel implementation of their pre-multiplication by a vector based on Application Specific Integrated Circuits (ASICs) and Field-Programmable Gate Arrays (FPGAs), incorporating Butterfly Circuits [DE].

Remark 3.3. We are likely to save flops for the approximation of the product $M\hat{B}$ if we use leverage scores [W14] (a.k.a. sampling probabilities [M11, Sections 3 and 5]).

3.9 Other basic families

There is a number of other interesting basic matrix families. According to [HMT11, Remark 4.6], “among the structured random matrices one of the strongest candidates involves sequences of random Givens rotations”. The sequences are defined in factored form, with two of the factors being the products of $n - 1$ Givens rotations each, two being permutation matrices, three matrices of diagonal scaling, and the DFT matrix Ω_n . The DFT factor makes the resulting matrices dense, but we can make them sparse by replacing that factor by an AF, ASF, APF, or ASPF matrix of recursion depth $d < \log_2(n)$. This would also decrease the number of flops involved in pre-multiplication of such a multiplier by a vector from order $n \log_2(n)$ to $1.5dn + O(n)$.

We can obtain new candidate families of efficient multipliers by replacing either or both of the Givens products with sparse matrices of Householder reflections matrices of the form $I_n - \frac{2\mathbf{h}\mathbf{h}^T}{\mathbf{h}^T\mathbf{h}}$ for fixed or random vectors \mathbf{h} (cf. [GL13, Section 5.1]). We can make these matrices sparse by choosing sparse vectors \mathbf{h} .

We can obtain a great variety of the families of multipliers promising to be efficient if we properly combine the matrices of basic families (i)–(iv) extended by the above matrices. This can be just linear combinations, but we can also use block representation as in the following real 2×2 block matrix $\frac{1}{\sqrt{n}} \begin{pmatrix} Z_1(\mathbf{u}) & Z_1(\mathbf{v}) \\ Z_1(\mathbf{v}) & -Z_1(\mathbf{u}) \end{pmatrix} D$ for two vectors \mathbf{u} and \mathbf{v} and a matrix D of primitive class 2.

The reader can find other useful families of multipliers in our Section 5. E.g., according to our tests in Section 5, it turned out to be efficient to use nonsingular well-conditioned (rather than unitary) diagonal factors in the definition of some of our basic matrix families.

4 Proof of Theorems 1.1, 1.3, and 1.4

4.1 Low-rank representation: proof

Theorem 4.1. (i) For an $m \times n$ input matrix M of rank $r \leq n \leq m$, its rank- r representation is given by the products $R(R^T R)^{-1} R^T M = Q(R)Q(R)^T M$ provided that R is an $n \times r$ matrix such that $\mathcal{R}(R) = \mathcal{R}(M)$ and that $Q(R)$ is a matrix obtained by means of column orthogonalization of R .

(ii) $\mathcal{R}(R) = \mathcal{R}(M)$, for $R = MB$ and an $n \times r$ matrix B , with probability 1 if B is Gaussian and

(iii) with a probability at least $1 - r/|S|$ if an $n \times r$ matrix B has i.i.d. random entries sampled uniformly from a finite set \mathcal{S} of cardinality $|S|$.

Proof. Readily verify part (i). Then note that $\mathcal{R}(MB) \subseteq \mathcal{R}(M)$, for an $n \times r$ multiplier B . Hence $\mathcal{R}(MB) = \mathcal{R}(M)$ if and only if $\text{rank}(MB) = r$, and therefore if and only if a multiplier B has full rank r .

Now parts (ii) and (iii) follow from Theorem B.1. □

Parts (i) and (ii) of Theorem 4.1 imply parts (i) of Theorems 1.3 and 1.4.

4.2 From low-rank representation to low-rank approximation: a basic step

Our extension of the above results to the proof of Theorems 1.1, 1.3, and 1.4 relies on our next lemma and theorem. Hereafter $\sigma_j(M)$ denotes the j th largest singular value of the matrix M (cf. Appendix A).

Lemma 4.1. (Cf. [GL13, Theorem 2.4.8].) *For an integer r and an $m \times n$ matrix M where $m \geq n > r > 0$, set to 0 the singular values $\sigma_j(M)$, for $j > r$, let M_r denote the resulting matrix, which is a closest rank- r approximation of M , and write $M = M_r + E$. Then*

$$\|E\| = \sigma_{r+1}(M) \text{ and } \|E\|_F^2 = \sum_{j=r+1}^n \sigma_j^2 \leq \sigma_{r+1}(M)^2(n-r).$$

Theorem 4.2. The error norm in terms of $\|(M_r B)^+\|$. Assume dealing with the matrices M and \tilde{M} of Algorithm 1.1, M_r and E of Lemma 4.1, and $B \in \mathbb{C}^{n \times l}$ of rank l . Let $\text{rank}(M_r B) = r$ and write $E' = EB$ and $\Delta = \|\tilde{M} - M\|$. Then

$$\|E'\|_F \leq \|B\|_F \|E\|_F \leq \|B\|_F \sigma_{r+1}(M) \sqrt{n-r} \quad (4.1)$$

and

$$|\Delta - \sigma_{r+1}(M)| \leq \sqrt{8} \|(M_r B)^+\| \|E'\|_F + O(\|E'\|_F^2). \quad (4.2)$$

Proof. Lemma 4.1 implies bound (4.1).

Now apply part (ii) of Theorem 4.1 for matrix M_r replacing M , recall that $\text{rank}(B) = l$, and obtain

$$Q(M_r B)Q(M_r B)^T M_r = M_r, \quad \mathcal{R}(Q(M_r B)) = \mathcal{R}(M_r B) = \mathcal{R}(M_r).$$

Therefore

$$Q(M_r B)Q(M_r B)^T M = Q(M_r B)Q(M_r B)^T M_r = M_r.$$

Consequently, $M - Q(M_r B)Q(M_r B)^T M = M - M_r = E$, and so (cf. Lemma 4.1)

$$\|M - Q(M_r B)Q(M_r B)^T M\| = \sigma_{r+1}(M). \quad (4.3)$$

Apply [PQY15, Corollary C.1], for $A = M_r B$ and E replaced by $E' = (M - M_r)B$, and obtain

$$\|Q(MB)Q(MB)^T - Q(M_r B)Q(M_r B)^T\| \leq \sqrt{8} \|(M_r B)^+\| \|E'\|_F + O(\|E'\|_F^2).$$

Combine this bound with (4.3) and obtain (4.2). \square

By combining parts (4.1) and (4.2) obtain

$$|\Delta - \sigma_{r+1}(M)| \leq \sqrt{8(n-r)} \sigma_{r+1}(M) \|B\|_F \|(M_r B)^+\| + O(\sigma_{r+1}^2(M)). \quad (4.4)$$

In our applications the product $\sigma_{r+1}(M)\|B\|_F$ is small, and so the value $|\Delta - \sigma_{r+1}(M)|$ is small unless the norm $\|(M_r B)^+\|$ is large.

Remark 4.1. The Power Scheme of increasing the output accuracy of Algorithm 1.1. See [RST09], [HMST11]. Define the Power Iterations $M_i = (M^T M)^i M$, $i = 1, 2, \dots$. Then $\sigma_j(M_i) = (\sigma_j(M))^{2i+1}$ for all i and j [HMT11, equation (4.5)]. Therefore, at a reasonable computational cost, one can dramatically decrease the ratio $\frac{\sigma_{r+1}(M)}{\sigma_r(M)}$ and thus decrease the bounds of Theorems 4.3 and 4.4 accordingly.

4.3 Proof of Theorem 1.1

If $\text{rank}(MB) = \text{rank}(Q) = r_- < r$, then $\text{rank}(\tilde{M}) \leq r_- < r$, $\Delta \geq \sigma_{r_-}(M)$, that is, not small since $\text{nrnk}(M) = r > r_-$, and so Algorithm 1.1 applied to M with the multiplier B outputs FAILURE. If $\text{rank}(MB) = r > \text{nrnk}(MB) = r_-$, then $\text{rank}(MB - E) = r_- < r$ for a small-norm perturbation matrix E . Hence $\Delta \geq \sigma_{r_-}(M) - O(\|E\|)$, and then again Algorithm 1.1 applied to M with the multiplier B outputs FAILURE. This proves the “only if” part of the claim of Theorem 1.1.

Now let $\text{nrnk}(MB) = r$ and assume that we scaled the matrix B so that $\|B\|_F = 1$. Then $\text{rank}(MB) = r$ (and so we can apply bound (4.4)), and furthermore $\text{nrnk}(M_r B) = \text{nrnk}(MB) = r$. Equation (4.4) implies that $\Delta \approx 8\sqrt{8(n-r)}\sigma_{r+1}\|(M_r B)^+\|$, which is a small positive value because $\text{nrnk}(M) = r$, and so the value $|\sigma_{r+1}|$ is small, and part “if” of Theorem 1.1 follows.

4.4 Detailed estimates for primal and dual low-rank approximation

The following theorem, proven in the next subsection, bounds the approximation errors and the probability of success of Algorithm 1.1 for $B \in \mathcal{G}^{n \times l}$. Together these bounds imply part (ii) of Theorem 1.3.

Theorem 4.3. *Suppose that Algorithm 1.1 has been applied to an $m \times n$ matrix M having numerical rank r and that the multiplier $B = G_{n,l}$ is an $n \times l$ Gaussian matrix.*

(i) *Then the algorithm outputs an approximation \tilde{M} of a matrix M by a rank- l matrix within the error norm bound Δ such that $|\Delta - \sigma_{r+1}(M)| \leq f\sigma_{r+1}(M) + O(\sigma_{r+1}^2(M))$ where $f = \sqrt{8(n-r)} \nu_{F,n,l} \nu_{r,l}^+ / \sigma_r(M)$ and $\nu_{F,n,l}$ and $\nu_{r,l}^+$ are random variables of Definition B.1.*

(ii) $\mathbb{E}(f) < \frac{1+\sqrt{n}+\sqrt{l}}{p\sigma_r(M)} e \sqrt{8(n-r)rl}$, for $p = l - r > 0$ and $e = 2.71828 \dots$.

Remark 4.2. $\sigma_{r+1}(M)$ is the optimal upper bound on the norm Δ , and the expected value $\mathbb{E}(f)$ is reasonably small even for $p = 1$. If $p = 0$, then $\mathbb{E}(f)$ is not defined, but the random variable Δ estimated in Theorem 4.3 is still likely to be reasonably close to $\sigma_{r+1}(M)$ (cf. part (ii) of Theorem B.3).

In Section 4.6 we prove the following elaboration upon dual Theorem 1.4; its upper bound on the output error norm of Algorithm 1.1 is a little smaller than the bound of primal Theorem 4.3.

Theorem 4.4. *Suppose that Algorithm 1.1, applied to a small-norm perturbation of an $m \times n$ factor-Gaussian matrix with expected rank $r < m$, uses an $n \times l$ multiplier B such that $\text{nrnk}(B) = l$ and $l \geq r$.*

(i) *Then the algorithm outputs a rank- l matrix \tilde{M} that approximates the matrix M within the error norm bound Δ such that $|\Delta - \sigma_{r+1}(M)| \leq f_d \sigma_{r+1}(M) + O(\sigma_{r+1}^2(M))$, where $f_d = \sqrt{8(n-r)l} \nu_{r,l}^+ \nu_{m,r}^+ \kappa(B)$, $\kappa(B) = \|B\| \|B^+\|$, and $\nu_{m,r}^+$ and $\nu_{r,l}^+$ are random variables of Definition B.1.*

(ii) $\mathbb{E}(f_d) < e^2 \sqrt{8(n-r)l} \kappa(B) \frac{r}{(m-r)p}$, for $p = l - r > 0$ and $e = 2.71828 \dots$.

Remark 4.3. The expected value $\mathbb{E}(\nu_{m,r}^+) = \frac{e\sqrt{r}}{m-r}$ converges to 0 as $m \rightarrow \infty$ provided that $r \ll m$. Consequently the expected value $\mathbb{E}(\Delta) = \sigma_{r+1}(M) \mathbb{E}(f_d)$ converges to the optimal value $\sigma_{r+1}(M)$ as $\frac{m}{r\sqrt{nl}} \rightarrow \infty$ provided that B is a well-conditioned matrix of full rank and that $1 \leq r < l \leq n \leq m$.

Remark 4.4. [HMT11, Theorem 10.8] also estimates the norm Δ , but our estimate in Theorem 4.3, in terms of random variables $\nu_{F,n,l}$ and $\nu_{r,l}^+$, is more compact, and our proof is distinct and shorter than one in [HMT11], which involves the proofs of [HMT11, Theorems 9.1, 10.4 and 10.6].

Remark 4.5. By virtue of Theorems B.1, $\text{rank}(M_r B) = r$ with probability 1 if the matrix B or M is Gaussian, which is the case of Theorems 4.3 and 4.4, and under the equation $\text{rank}(M_r B) = r$ we proved bound (4.4).

In the next two subsections we deduce reasonable bounds on the norm $\|(M_r B)^+\|$ in both cases where M is a fixed matrix and B is a Gaussian matrix and where B is fixed matrix and M is a factor Gaussian matrix (cf. Theorems 4.5 and 4.6). The bounds imply Theorems 4.3 and 4.4.

4.5 Primal theorem: completion of the proof

Theorem 4.5. *For $M \in \mathbb{R}^{m \times n}$, $B \in \mathcal{G}^{m \times l}$, and $\nu_{r,l}^+$ of Definition B.1, it holds that*

$$\|(M_r B)^+\| \leq \nu_{r,l}^+ / \sigma_r(M). \quad (4.5)$$

Proof. Let $M_r = S_r \Sigma_r T_r^T$ be compact SVD.

By applying Lemma B.1, deduce that $T_r^T B$ is a $r \times l$ Gaussian matrix.

Denote it $G_{r,l}$ and obtain $M_r B = S_r \Sigma_r T_r^T B = S_r \Sigma_r G_{r,l}$.

Write $H = \Sigma_r G_{r,l}$ and let $H = S_H \Sigma_H T_H^T$ be compact SVD where S_H is a $r \times r$ unitary matrix.

It follows that $S = S_r S_H$ is an $m \times r$ unitary matrix.

Hence $M_r B = S \Sigma_H T_H^T$ and $(M_r B)^+ = T_H (\Sigma_H)^+ S^T$ are compact SVDs of the matrices $M_r B$ and $(M_r B)^+$, respectively.

Therefore $\|(M_r B)^+\| = \|(\Sigma_H)^+\| = \|(\Sigma_r G_{r,l})^+\| \leq \|G_{r,l}^+\| \|\Sigma_r^{-1}\|$.

Substitute $\|G_{r,l}^+\| = \nu_{r,l}^+$ and $\|\Sigma_r^{-1}\| = 1/\sigma_r(M)$ and obtain the theorem. \square

Combine bounds (4.4), (4.5), and equation $\|B\|_F = \nu_{F,n,l}$ and obtain part (i) of Theorem 4.3. Combine that part with parts (ii) of Theorem B.2 and (iii) of Theorem B.3 and obtain part (ii) of Theorem 4.3.

4.6 Dual theorem: completion of the proof

Theorem 4.6. *Suppose that $U \in \mathbb{R}^{m \times r}$, $V \in \mathcal{G}^{r \times n}$, $\text{rank}(U) = r \leq \min\{m, n\}$, $M = UV$, and B is a well-conditioned $n \times l$ matrix of full rank l such that $m \geq n > l \geq r$ and $\|B\|_F = 1$. Then*

$$\|(MB)^+\| \leq \|U^+\| \nu_{r,l}^+ \|B^+\|. \quad (4.6)$$

If in addition $U \in \mathcal{G}^{m \times r}$, that is, if M is an $m \times n$ factor-Gaussian matrix with expected rank r , then

$$\|(MB)^+\| \leq \nu_{m,r}^+ \nu_{r,l}^+ \|B^+\|. \quad (4.7)$$

Proof. Combine compact SVDs $U = S_U \Sigma_U T_U^T$ and $B = S_B \Sigma_B T_B^T$ and obtain $UVB = S_U \Sigma_U T_U^T V S_B \Sigma_B T_B^T$. Here $U, V, B, S_U, \Sigma_U, T_U, S_B, \Sigma_B$, and T_B are matrices of the sizes $m \times r, r \times n, n \times l, m \times r, r \times r, r \times r, n \times l, l \times l$, and $l \times l$, respectively.

Now observe that $G_{r,l} = T_U^T V S_B$ is a $r \times l$ Gaussian matrix, by virtue of Lemma B.1 (since $V = G_{r,n}$ is a Gaussian matrix). Therefore $UVB = S_U F T_B^T$, for $F = \Sigma_U G_{r,l} \Sigma_B$.

Let $F = S_F \Sigma_F T_F^T$ denote compact SVD where $\Sigma_F = \text{diag}(\sigma_j(F))_{j=1}^r$ and S_F and T_F^T are unitary matrices of sizes $r \times r$ and $r \times l$, respectively.

Both products $S_U S_F \in \mathbb{R}^{m \times r}$ and $T_F^T T_B^T \in \mathbb{R}^{r \times l}$ are unitary matrices, and we obtain compact SVD $MB = UVB = S_{MB} \Sigma_{MB} T_{MB}^T$ where $S_{MB} = S_U S_F$, $\Sigma_{MB} = \Sigma_F$, and $T_{MB}^T = T_F^T T_B^T$. Therefore

$$\|(MB)^+\| = \|\Sigma_{MB}^+\| = \|\Sigma_F^+\| = \|F^+\|.$$

Note that $F^+ = \Sigma_B^{-1} G_{r,l}^+ \Sigma_U^{-1}$ because Σ_B and Σ_U are square nonsingular diagonal matrices. It follows that

$$\|F^+\| \leq \|\Sigma_B^{-1}\| \|G_{r,l}^+\| \|\Sigma_U^{-1}\| = \|B^+\| \nu_{r,l}^+ \|V^+\|.$$

Consequently $\|(MB)^+\| = \|\Sigma_F^+\| \leq \|U^+\| \nu_{r,l}^+ \|B^+\|$. Now (4.6) follows and implies (4.7). \square

Combine (4.4), (4.6) and $\|B\|_F \leq \|B\| \sqrt{l}$ and obtain Theorem 4.4 provided that M is a factor-Gaussian matrix UV with expected rank r . Apply Theorem A.1 to extend the results to the case where $M = UV + E$ and the norm $\|E\|$ is small, completing the proof of Theorem 4.4.

Remark 4.6. *If $U \in \mathcal{G}^{m \times r}$, for $m - r \geq 4$, then it is likely that $\text{nrank}(U) = r$ by virtue of Theorem B.3, and our proof of bound (4.6) applies even if we assume that $\text{nrank}(U) = r$ rather than $U \in \mathcal{G}^{m \times r}$.*

5 Numerical Tests

Numerical experiments have been performed by Xiaodong Yan for Tables 5.2–5.4 and by John Svadlenka and Liang Zhao for the other tables. The tests have been run by using MATLAB in the Graduate Center of the City University of New York on a Dell computer with the Intel Core 2 2.50 GHz processor and 4G memory running Windows 7, and in particular the standard normal distribution function `randn` of MATLAB has been applied in order to generate Gaussian matrices.

We calculated the ξ -rank, i.e., the number of singular values exceeding ξ , by applying the MATLAB function "svd()". We have set $\xi = 10^{-5}$ in Sections 5.1 and 5.2 and $\xi = 10^{-6}$ in Section 5.3.

5.1 Tests for inputs generated via SVD

In the tests of this subsection we generated $n \times n$ input matrices M by extending the customary recipes of [H02, Section 28.3]. Namely, we first generated matrices S_M and T_M by means of the orthogonalization of $n \times n$ Gaussian matrices. Then we defined $n \times n$ matrices M by their compact SVDs, $M = S_M \Sigma_M T_M^T$, for $\Sigma_M = \text{diag}(\sigma_j)_{j=1}^n$; $\sigma_j = 1/j$, $j = 1, \dots, r$, $\sigma_j = 10^{-10}$, $j = r + 1, \dots, n$, and $n = 256, 512, 1024$. (Hence $\|M\| = 1$ and $\kappa(M) = \|M\| \|M^{-1}\| = 10^{10}$.)

Table 5.1 shows the average output error norm Δ over 1000 tests of Algorithm 1.1 applied to these matrices M for each pair of n and r , $n = 256, 512, 1024$, $r = 8, 32$, and each of the following three groups of multipliers: 3-AH multipliers, 3-ASPH multipliers, both defined by Hadamard recursion (3.3), for $d = 3$, and dense multipliers $B = B(\pm 1, 0)$ having i.i.d. entries ± 1 and 0, each value chosen with probability $1/3$.

Table 5.1: Error norms for SVD-generated inputs and 3-AH, 3-ASPH, and $B(\pm 1, 0)$ multipliers

n	r	3-AH	3-ASPH	$B(\pm 1, 0)$
256	8	2.25e-08	2.70e-08	2.52e-08
256	32	5.95e-08	1.47e-07	3.19e-08
512	8	4.80e-08	2.22e-07	4.76e-08
512	32	6.22e-08	8.91e-08	6.39e-08
1024	8	5.65e-08	2.86e-08	1.25e-08
1024	32	1.94e-07	5.33e-08	4.72e-08

Tables 5.2–5.4 show the mean and maximal values of such an error norm in the case of (a) real Gaussian multipliers B and dense real Gaussian subcirculant multipliers B of Section 3.3, for $q = n$, each defined by its first column filled with either (b) i.i.d. Gaussian variables or (c) random variables ± 1 . Here and hereafter in this section we assigned each random signs $+$ or $-$ with probability 0.5.

Table 5.2: Error norms for SVD-generated inputs and Gaussian multipliers

r	n	mean	max
8	256	7.54×10^{-8}	1.75×10^{-5}
8	512	4.57×10^{-8}	5.88×10^{-6}
8	1024	1.03×10^{-7}	3.93×10^{-5}
32	256	5.41×10^{-8}	3.52×10^{-6}
32	512	1.75×10^{-7}	5.57×10^{-5}
32	1024	1.79×10^{-7}	3.36×10^{-5}

Table 5.3: Error norms for SVD-generated inputs and Gaussian subcirculant multipliers

r	n	mean	max
8	256	3.24×10^{-8}	2.66×10^{-6}
8	512	5.58×10^{-8}	1.14×10^{-5}
8	1024	1.03×10^{-7}	1.22×10^{-5}
32	256	1.12×10^{-7}	3.42×10^{-5}
32	512	1.38×10^{-7}	3.87×10^{-5}
32	1024	1.18×10^{-7}	1.84×10^{-5}

Table 5.5 displays the average error norms in the case of multipliers B of eight kinds defined below, all generated from the following Basic Sets 1, 2 and 3 of $n \times n$ multipliers:

Basic Set 1: 3-APF multipliers defined by three Fourier recursive steps of equation (3.3), for $d = 3$, with no scaling, but with a random column permutation.

Basic Set 2: Sparse real circulant matrices $Z_1(\mathbf{v})$ of family (ii) of Section 3.3, for $q = 10$, whose first column vectors \mathbf{v} have been filled with zeros, except for ten random coordinates filled with random integers ± 1 .

Basic Set 3: Sum of two scaled inverse bidiagonal matrices. We first filled the main diagonals of both

Table 5.4: Error norms for SVD-generated inputs and random subcirculant multipliers filled with ± 1

r	n	mean	max
8	256	7.70×10^{-9}	2.21×10^{-7}
8	512	1.10×10^{-8}	2.21×10^{-7}
8	1024	1.69×10^{-8}	4.15×10^{-7}
32	256	1.51×10^{-8}	3.05×10^{-7}
32	512	2.11×10^{-8}	3.60×10^{-7}
32	1024	3.21×10^{-8}	5.61×10^{-7}

matrices with the integer 101 and their first subdiagonals with ± 1 . Then we multiplied each matrix by a diagonal matrix $\text{diag}(\pm 2^{b_i})$, where b_i were random integers uniformly chosen from 0 to 3.

For multipliers B we used the $n \times r$ western (leftmost) blocks of $n \times n$ matrices of the following classes:

1. a matrix from Basic Set 1;
2. a matrix from Basic Set 2;
3. a matrix from Basic Set 3;
4. the product of two matrices of Basic Set 1;
5. the product of two matrices of Basic Set 2;
6. the product of two matrices of Basic Set 3;
7. the sum of two matrices of Basic Sets 1 and 3, and
8. the sum of two matrices of Basic Sets 2 and 3.

The tests produced the results similar to the ones of Tables 5.1–5.4.

In sum, for all classes of input pairs M and B and all pairs of integers n and r , Algorithm 1.1 with our preprocessing has consistently output approximations to rank- r input matrices with the average error norms ranged from 10^{-7} or 10^{-8} to about 10^{-9} in all our tests.

Table 5.5: Error norms for SVD-generated inputs and multipliers of eight classes

n	r	class 1	class 2	class 3	class 4	class 5	class 6	class 7	class 8
256	8	5.94e-09	4.35e-08	2.64e-08	2.20e-08	7.73e-07	5.15e-09	4.08e-09	2.10e-09
256	32	2.40e-08	2.55e-09	8.23e-08	1.58e-08	4.58e-09	1.36e-08	2.26e-09	8.83e-09
512	8	1.11e-08	8.01e-09	2.36e-09	7.48e-09	1.53e-08	8.15e-09	1.39e-08	3.86e-09
512	32	1.61e-08	4.81e-09	1.61e-08	2.83e-09	2.35e-08	3.48e-08	2.25e-08	1.67e-08
1024	8	5.40e-09	3.44e-09	6.82e-08	4.39e-08	1.20e-08	4.44e-09	2.68e-09	4.30e-09
1024	32	2.18e-08	2.03e-08	8.72e-08	2.77e-08	3.15e-08	7.99e-09	9.64e-09	1.49e-08

We summarize the results of the tests of this subsection for $n = 1024$ and $r = 8, 32$ in Figure 1.

Figure 1: Error norms in the tests of Section 5.1

5.2 Tests for inputs generated via the discretization of a Laplacian operator and via the approximation of an inverse finite-difference operator

Next we present the test results for Algorithm 1.1 applied to input matrices for computational problems of two kinds, both replicated from [HMT11], namely, the matrices of

- (i) the discretized single-layer Laplacian operator and
- (ii) the approximation of the inverse of a finite-difference operator.

Input matrices (i). We considered the Laplacian operator $[S\sigma](x) = c \int_{\Gamma_1} \log|x-y|\sigma(y)dy, x \in \Gamma_2$, from [HMT11, Section 7.1], for two contours $\Gamma_1 = C(0,1)$ and $\Gamma_2 = C(0,2)$ on the complex plane. Its discretization defines an $n \times n$ matrix $M = (m_{ij})_{i,j=1}^n$ where $m_{i,j} = c \int_{\Gamma_{1,j}} \log|2\omega^i - y|dy$ for a constant c such that $\|M\| = 1$ and for the arc $\Gamma_{1,j}$ of the contour Γ_1 defined by the angles in $[\frac{2j\pi}{n}, \frac{2(j+1)\pi}{n}]$. We applied Algorithm 1.1 supported by three iterations of the Power Scheme of Remark 4.1 and used with multipliers B being the $n \times r$ leftmost submatrices of $n \times n$ matrices of the following five classes:

- Gaussian multipliers,
- Gaussian Toeplitz multipliers $T = (t_{i-j})_{i=0}^{n-1}$ for i.i.d. Gaussian variables $t_{1-n}, \dots, t_{-1}, t_0, t_1, \dots, t_{n-1}$.
- Gaussian circulant multipliers $\sum_{i=0}^{n-1} v_i Z_1^i$, for i.i.d. Gaussian variables v_0, \dots, v_{n-1} and the unit circular matrix Z_1 of Section 3.2.
- Abridged permuted Fourier (3-APF) multipliers, and
- Abridged permuted Hadamard (3-APH) multipliers.

As in the previous subsection, we defined each 3-APF and 3-APH matrix by applying three recursive steps of equation (3.3) followed by a single random column permutation.

We applied Algorithm 1.1 with multipliers of all five listed classes. For each setting we repeated the test 1000 times and calculated the mean and standard deviation of the error norm $\|\tilde{M} - M\|$.

Table 5.6: Low-rank approximation of Laplacian matrices

n	multiplier	r	mean	std
200	Gaussian	3.00	1.58e-05	1.24e-05
200	Toeplitz	3.00	1.83e-05	7.05e-06
200	Circulant	3.00	3.14e-05	2.30e-05
200	3-APF	3.00	8.50e-06	5.15e-15
200	3-APH	3.00	2.18e-05	6.48e-14
400	Gaussian	3.00	1.53e-05	1.37e-06
400	Toeplitz	3.00	1.82e-05	1.59e-05
400	Circulant	3.00	4.37e-05	3.94e-05
400	3-APF	3.00	8.33e-06	1.02e-14
400	3-APH	3.00	2.18e-05	9.08e-14
2000	Gaussian	3.00	2.10e-05	2.28e-05
2000	Toeplitz	3.00	2.02e-05	1.42e-05
2000	Circulant	3.00	6.23e-05	7.62e-05
2000	3-APF	3.00	1.31e-05	6.16e-14
2000	3-APH	3.00	2.11e-05	4.49e-12
4000	Gaussian	3.00	2.18e-05	3.17e-05
4000	Toeplitz	3.00	2.52e-05	3.64e-05
4000	Circulant	3.00	8.98e-05	8.27e-05
4000	3-APF	3.00	5.69e-05	1.28e-13
4000	3-APH	3.00	3.17e-05	8.64e-12

Input matrices (ii). We similarly applied Algorithm 1.1 to the input matrix M being the inverse of a large sparse matrix obtained from a finite-difference operator from [HMT11, Section 7.2] and observed similar results with all structured and Gaussian multipliers.

We performed 1000 tests for every class of pairs of $n \times n$ or $m \times n$ matrices of classes (i) or (ii), respectively, and $n \times r$ multipliers for every fixed triple of m , n , and r or pair of n and r .

Tables 5.6 and 5.7 display the resulting data for the mean values and standard deviation of the error norms, and we summarize the results of the tests of this subsection in Figure 2.

Table 5.7: Low-rank approximation of the matrices of discretized finite-difference operator

m	n	multiplier	r	mean	std
88	160	Gaussian	5.00	1.53e-05	1.03e-05
88	160	Toeplitz	5.00	1.37e-05	1.17e-05
88	160	Circulant	5.00	2.79e-05	2.33e-05
88	160	3-APF	5.00	4.84e-04	2.94e-14
88	160	3-APH	5.00	4.84e-04	5.76e-14
208	400	Gaussian	43.00	4.02e-05	1.05e-05
208	400	Toeplitz	43.00	8.19e-05	1.63e-05
208	400	Circulant	43.00	8.72e-05	2.09e-05
208	400	3-APF	43.00	1.24e-04	2.40e-13
208	400	3-APH	43.00	1.29e-04	4.62e-13
408	800	Gaussian	64.00	6.09e-05	1.75e-05
408	800	Toeplitz	64.00	1.07e-04	2.67e-05
408	800	Circulant	64.00	1.04e-04	2.67e-05
408	800	3-APF	64.00	1.84e-04	6.42e-12
408	800	3-APH	64.00	1.38e-04	8.65e-12

Figure 2: Error norms in the tests of Section 5.1

5.3 Tests with additional classes of multipliers

In this subsection we display the mean values and standard deviations of the error norms observed when we repeated the tests of the two previous subsections for the same three classes of input matrices (that is, SVD-generated, Laplacian, and matrices obtained by discretization of finite difference operators), but now we applied Algorithm 1.1 with seventeen additional classes of multipliers (besides its control application with Gaussian multipliers).

We tested Algorithm 1.1 applied to 1024×1024 SVD-generated input matrices having numerical nullity $r = 32$, to 400×400 Laplacian input matrices having numerical nullity $r = 3$, and to 408×800 matrices having numerical nullity $r = 64$ and representing finite-difference inputs.

Then again we repeated the tests 1000 times for each class of input matrices and each size of an input and a multiplier, and we display the resulting average error norms in Table 5.3 and Figures 3–5.

We used multipliers defined as the seventeen sums of $n \times r$ matrices of the following basic families:

- 3-ASPH matrices
- 3-APH matrices
- Inverses of bidiagonal matrices
- Random permutation matrices

Here every 3-APH matrix has been defined by three Hadamard's recursive steps (3.1) followed by random permutation. Every 3-ASPH matrix has been defined similarly, but also random scaling has been applied with a diagonal matrix $D = \text{diag}(d_i)_{i=1}^n$ having the values of random i.i.d. variables d_i uniformly chosen from the set $\{1/4, 1/2, 1, 2, 4\}$.

We permuted all inverses of bidiagonal matrices except for Class 5 of multipliers.

Describing our multipliers we use the following acronyms and abbreviations: “IBD” for “the inverse of a bidiagonal”, “MD” for “the main diagonal”, “SB” for “subdiagonal”, and “SP” for “superdiagonal”. We write “MD i ”, “ k th SB i ” and “ k th SP i ” in order to denote that the main diagonal, the k th subdiagonal, or the k th superdiagonal of a bidiagonal matrix, respectively, was filled with the integer i .

- Class 0: Gaussian
- Class 1: Sum of a 3-ASPH and two IBD matrices:
B1 with MD−1 and 2nd SB−1 and B2 with MD+1 and 1st SP+1
- Class 2: Sum of a 3-ASPH and two IBD matrices:
B1 with MD+1 and 2nd SB−1 and B2 with MD+1 and 1st SP−1
- Class 3: Sum of a 3-ASPH and two IBD matrices:
B1 with MD+1 and 1st SB−1 and B2 with MD +1 and 1st SP−1
- Class 4: Sum of a 3-ASPH and two IBD matrices:
B1 with MD+1 and 1st SB+1 and B2 with MD+1 and 1st SP−1
- Class 5: Sum of a 3-ASPH and two IBD matrices:
B1 with MD+1 and 1st SB+1 and B2 with MD+1 and 1st SP−1
- Class 6: Sum of a 3-ASPH and three IBD matrices:
B1 with MD−1 and 2nd SB−1, B2 with MD+1 and 1st SP+1 and B3 with MD+1 and 9th SB+1
- Class 7: Sum of a 3-ASPH and three IBD matrices:
B1 with MD+1 and 2nd SB−1, B2 with MD+1 and 1st SP−1, and B3 with MD+1 and 8th SP+1
- Class 8: Sum of a 3-ASPH and three IBD matrices:
B1 with MD+1 and 1st SB−1, B2 with MD+1 and 1st SP−1, and B3 with MD+1 and 4th SB+1
- Class 9: Sum of a 3-ASPH and three IBD matrices:
B1 with MD+1 and 1st SB+1, B2 with MD+1 and 1st SP−1, and B3 with MD−1 and 3rd SP+1
- Class 10: Sum of three IBD matrices:
B1 with MD+1 and 1st SB+1, B2 with MD+1 and 1st SP−1, and B3 with MD−1 and 3rd SP+1
- Class 11: Sum of a 3-APH and three IBD matrices:
B1 with MD+1 and 2nd SB−1, B2 with MD+1 and 1st SP−1, and B3 with MD+1 and 8th SP+1
- Class 12: Sum of a 3-APH and two IBD matrices:
B1 with MD+1 and 1st SB−1 and B2 with MD+1 and 1st SP−1
- Class 13: Sum of a 3-ASPH and a permutation matrix
- Class 14: Sum of a 3-ASPH and two permutation matrices
- Class 15: Sum of a 3-ASPH and three permutation matrices
- Class 16: Sum of a 3-APH and three permutation matrices
- Class 17: Sum of a 3-APH and two permutation matrices

The tests show quite accurate outputs even where we applied Algorithm 1.1 with very sparse multipliers of classes 13–17.

Figure 3: Relative Error Norm For SVD Matrices

Figure 4: Relative Error Norm For Lapacian Matrices

Figure 5: Relative Error Norm For Finite-Difference Matrices

	SVD Matrices		Laplacian Matrices		Finite Difference Matrices	
Class No.	Mean	Std	Mean	Std	Mean	Std
Class 0	3.54E-09	3.28E-09	4.10E-14	2.43E-13	1.61E-06	1.35E-06
Class 0	1.07E-08	3.82E-09	2.05E-13	1.62E-13	4.58E-06	9.93E-07
Class 1	1.16E-08	6.62E-09	6.07E-13	5.20E-13	4.67E-06	1.04E-06
Class 2	1.23E-08	5.84E-09	1.69E-13	1.34E-13	4.52E-06	1.01E-06
Class 3	1.25E-08	1.07E-08	2.46E-13	3.44E-13	4.72E-06	9.52E-07
Class 4	1.13E-08	6.09E-09	1.93E-13	1.48E-13	4.38E-06	8.64E-07
Class 5	1.12E-08	8.79E-09	9.25E-13	2.64E-12	5.12E-06	1.29E-06
Class 6	1.16E-08	7.42E-09	5.51E-13	5.35E-13	4.79E-06	1.12E-06
Class 7	1.33E-08	1.00E-08	1.98E-13	1.30E-13	4.60E-06	9.52E-07
Class 8	1.08E-08	4.81E-09	2.09E-13	3.60E-13	4.47E-06	8.57E-07
Class 9	1.18E-08	5.51E-09	1.87E-13	1.77E-13	4.63E-06	9.28E-07
Class 10	1.18E-08	6.23E-09	1.78E-13	1.42E-13	4.55E-06	9.08E-07
Class 11	1.28E-08	1.40E-08	2.33E-13	3.44E-13	4.49E-06	9.67E-07
Class 12	1.43E-08	1.87E-08	1.78E-13	1.61E-13	4.74E-06	1.19E-06
Class 13	1.22E-08	1.26E-08	2.21E-13	2.83E-13	4.75E-06	1.14E-06
Class 14	1.51E-08	1.18E-08	3.57E-13	9.27E-13	4.61E-06	1.08E-06
Class 15	1.19E-08	6.93E-09	2.24E-13	1.76E-13	4.74E-06	1.09E-06
Class 16	1.26E-08	1.16E-08	2.15E-13	1.70E-13	4.59E-06	1.12E-06
Class 17	1.31E-08	1.18E-08	1.25E-14	5.16E-14	1.83E-06	1.55E-06

Table 5.8: Relative Error Norm with Superfast Multipliers

6 Conclusions: Three Extensions

Our duality techniques for the average inputs can be extended to the acceleration of various matrix computations. In this concluding section we describe extensions to three highly important areas.

6.1 Acceleration of the computation for Least Squares Regression (LSR)

We first recall the following fundamental problem of matrix computations (cf. [GL13]):

Problem 6.1. Least Squares Solution of an Overdetermined Linear System of Equations. *Given two integers m and d such that $1 \leq d < m$, a matrix $A \in \mathbb{R}^{m \times d}$, and a vector $\mathbf{b} \in \mathbb{R}^m$, compute a vector \mathbf{x} that minimizes the norm $\|A\mathbf{x} - \mathbf{b}\|$.*

If a matrix A has full rank n , then unique solution is given by the vector $\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b}$, satisfying the linear system of normal equations $A^T A \mathbf{x} = A^T \mathbf{b}$. Otherwise solution is not unique, and a solution \mathbf{x} having the minimum norm is given by the vector $A^+ \mathbf{b}$. In the important case where $m \gg d$ and an approximate solution is acceptable, Sarlós in [S06] proposed to simplify the computations as follows:

Algorithm 6.1. Least Squares Regression (LSR).

INITIALIZATION: *Fix an integer k such that $1 \leq k \ll m$.*

COMPUTATIONS: 1. *Generate a scaled $k \times m$ Gaussian matrix F .*

2. *Compute the matrix FA and the vector $F\mathbf{b}$.*

3. *Output a solution $\tilde{\mathbf{x}}$ to the compressed Problem 6.1 where the matrix A and the vector \mathbf{b} are replaced by the matrix FA and the vector $F\mathbf{b}$, respectively.*

Now write $M = (A \mid \mathbf{b})$ and $\mathbf{y} = \begin{pmatrix} \mathbf{x} \\ -1 \end{pmatrix}$ and compare the error norms $\|FM\tilde{\mathbf{y}}\| = \|FA\tilde{\mathbf{x}} - F\mathbf{b}\|$ (of the output $\tilde{\mathbf{x}}$ of the latter algorithm) and $\|M\mathbf{y}\| = \|A\mathbf{x} - \mathbf{b}\|$ (of the solution \mathbf{x} of the original Problem 6.1).

Theorem 6.1. [W14, Theorem 2.3]. *Suppose that we are given two tolerance values δ and ξ , $0 < \delta < 1$ and $0 < \xi < 1$, three integers k , m and d such that $1 \leq d < m$ and*

$$k = (d + \log(1/\delta)\xi^{-2})\theta,$$

for a certain constant θ , and a matrix $G_{k,m} \in \mathcal{G}^{k \times m}$. Then, with a probability at least $1 - \delta$, it holds that

$$(1 - \xi)\|M\mathbf{y}\| \leq \frac{1}{\sqrt{k}}\|G_{k,m}M\mathbf{y}\| \leq (1 + \xi)\|M\mathbf{y}\|$$

for all matrices $M \in \mathbb{R}^{m \times (d+1)}$ and all vectors $\mathbf{y} = (y_i)_{i=0}^d \in \mathbb{R}^{d+1}$ normalized so that $y_d = -1$.

The theorem implies that with a probability at least $1 - \delta$, Algorithm 6.1 outputs an approximate solution to Problem 6.1 within the error norm bound ξ provided that $k = (d + \log(1/\delta)\xi^{-2})\theta$ and $F = \frac{1}{\sqrt{k}}G_{k,m}$.⁷

For $m \gg k$, the computational cost of performing the algorithm for approximate solution dramatically decreases versus the cost of computing exact solution, but can still be prohibitively high at the stage of computing the matrix product FM . In a number of papers the former cost has been dramatically decreased further by means of replacing a multiplier $F = \frac{1}{\sqrt{k}}G_{k,m}$ with various random sparse and structured matrices (see [W14, Section 2.1]), for which the bound of Theorem 6.1 still holds for all matrices $M \in \mathbb{R}^{m \times (d+1)}$, although at the expense of increasing significantly the dimension k .

Can we achieve similar progress without such an increase? The following theorem provides positive answer in the case where M is the average matrix in $\mathbb{R}^{m \times (d+1)}$ under the Gaussian probability distribution:

Theorem 6.2. Dual LSR. *The bound of Theorem 6.1 holds with a probability at least $1 - \delta$ where $\sqrt{k} M \in \mathcal{G}^{m \times (d+1)}$ and $F \in \mathbb{R}^{k \times m}$ is an orthogonal matrix.*

⁷Such approximate solutions serve as preprocessors for practical implementation of numerical linear algebra algorithms for Problem 6.1 of least squares computation [M11, Section 4.5], [RT08], [AMT10].

Proof. Theorem 6.1 has been proven in [W14, Section 2] in the case where $\sqrt{k} FM \in \mathcal{G}^{k \times (d+1)}$. This is the case where $\sqrt{k} F \in \mathcal{G}^{k \times m}$ and M is an orthogonal $m \times (d+1)$ matrix, but is also the case under the assumptions of Theorem 6.2, by virtue of Lemma B.1. \square

Theorem 6.2 supports the computation of LSR for any orthogonal multiplier F (e.g., a Hadamard’s scaled multiplier or a CountSketch multiplier from the data stream literature [W14, Section 2.1], [CCF04], [TZ12]) and for an input matrix $M \in \mathbb{R}^{m \times (d+1)}$, average under the Gaussian probability distribution.

It follows that Algorithm 6.1 can fail only for a narrow class of pairs F and M where F denotes orthogonal matrices in $\mathbb{R}^{k \times m}$ and M denotes matrices in $\mathbb{R}^{m \times (d+1)}$, and even in the case of failure we can still have good chances to succeed by using heuristic recipes of our Section 2.

6.2 Acceleration of the Fast Multipole Method (FMM) for the Average HSS matrix

Next we point out how our duality approach can accelerate a fundamental application of the FMM to multiplication by a vector of HSS matrix ⁸ defined by its average case generators. This preliminary sketch (together with its further extension to the CG algorithms in the next subsection) should demonstrate new chances for the expansion of the large application area of low-rank approximation.

We first recall that HSS matrices naturally extend the class of banded matrices and their inverses, are closely linked to FMM, have been intensively studied for decades (cf. [CGR88], [GR87], [T00], [BGH03], [GH03], [VVG05], [VVM07/08], [B10], [X12], [XXG12], [EGH13], [X13], [XXCB14], and the bibliography therein), and are highly and increasingly popular.

Definition 6.1. (Cf. [MRT05].) *With each diagonal block of a block matrix associate its complement in its block column, that is, the union of the pair of the maximal sub- and super-diagonal blocks in that column block, and call this complement a neutered block column.*

Definition 6.2. (Cf. [X12], [X13], [XXCB14].)

- A block matrix M of size $m \times n$ is called a r -HSS matrix, for a positive integer r ,
- (i) if all diagonal blocks of this matrix consist of $O((m+n)r)$ entries overall and
 - (ii) if r is the maximum rank of its neutered blocks.

Remark 6.1. Many authors work with (l, u) -HSS rather than r -HSS matrices M where l and u are the maximum ranks of the sub- and super-block-diagonal blocks, respectively. The (l, u) -HSS and r -HSS matrices are closely related. If a neutered block column N is the union of a sub-block-diagonal block B_- and a super-block-diagonal block B_+ , then $\text{rank}(N) \leq \text{rank}(B_-) + \text{rank}(B_+)$, and so an (l, u) -HSS matrix is a r -HSS matrix, for $r \leq l + u$, while clearly a r -HSS matrix is a (r, r) -HSS matrix.

The FMM exploits the r -HSS structure of a matrix as s (cf. [VVM07/08], [B10], [EGH13]):

- (i) Cover all off-block-diagonal entries with a set of non-overlapping neutered block columns.
- (ii) Express every neutered block column N of this set as the product FH of two generator matrices, F of size $h \times r$ and H of size $r \times k$. Call the pair $\{F, H\}$ a length r generator of the neutered block column N .
- (iii) Multiply readily the matrix M by a vector by separately multiplying generators and diagonal blocks by subvectors, which involves $O((m+n)r)$ flops overall, and
- (iv) in a more advanced application of FMM solve a nonsingular r -HSS linear system of n equations by using $O(nr \log^2(n))$ flops under some mild additional assumptions on the input.

This approach is readily extended to the same operations with (r, ξ) -HSS matrices, that is, matrices approximated by r -HSS matrices within a perturbation norm bound ξ where a positive tolerance ξ is small in context (e.g., is the unit round-off). Likewise, one defines an (r, ξ) -HSS representation and (r, ξ) -generators.

(r, ξ) -HSS matrices (for r small in context) appear routinely in matrix computations, and computations with such matrices are performed efficiently by using the above techniques.

In many applications of the FMM (cf., e.g., [BGP05], [VVVF10]) stage (ii) is omitted because short generators for all neutered block columns are readily available, but this is not the case in other important applications (cf. [XXG12], [XXCB14], [P15], and our Section 6.3). The computation of such generators is

⁸Here and hereafter “HSS” stands for “hierarchically semiseparable”.

precisely the low-rank approximation of the neutered block columns, which turns out to be the bottleneck stage of FMM in these applications.

Indeed apply random sampling Algorithm 1.1 at this stage with Gaussian multipliers. Multiplication of a $q \times h$ matrix by $h \times r$ Gaussian matrix requires $(2h - 1)qr$ flops, while standard HSS-representation of $n \times n$ HSS matrix includes $q \times h$ neutered block columns for $q \approx m/2$ and $h \approx n/2$. In this case the cost of computing an r -HSS representation of the matrix M is at least of order mnr . For large integers m and n , this greatly exceeds the above estimate of $O((m + n)r)$ flops at the other stages of the computations.

Alternative customary techniques for low-rank approximation rely on computing SVD or rank-revealing factorization of an input matrix and are at least as costly as the computations by means of random sampling.

Can we alleviate such a problem? Yes, we can accelerate randomized computation of the generators for the average neutered block columns involved into a desired (r, ξ) -HSS representation, by applying our recipes of Section 2 and multipliers of Section 3. The papers [XXG12] and [XXCB14] have succeeded by means of ad hoc application of random Toeplitz multipliers. Now our Dual Theorem 4.4 backs up that success, and our recipes of Section 2 enable further simplification of such computations.

6.3 Acceleration of the Conjugate Gradient (CG) algorithms

We recall that a real $n \times n$ matrix M and a linear system of n equations $M\mathbf{x} = \mathbf{b}$ are said to be *symmetric positive definite*⁹ if $M = V^T V$ for a nonsingular matrix V [GL13] and use the following concept:

Definition 6.3. *An $n \times n$ matrix M is (r, ξ) -concentrated if the set of its singular values is clustered (within a small tolerance ξ) about at most $r + 1$ values. Such a matrix is strongly (r, ξ) -concentrated if this set contains a cluster of at least $n - r$ singular values, each counted with its multiplicity.*

The following two facts imply the efficiency of the CG method:

- (i) Given a spd linear system of equations $M\mathbf{x} = \mathbf{b}$ whose matrix M is (r, ξ) -concentrated, the CG algorithms converge to its solution within a error norm in $O(\xi)$ in at most r iterations [A94], [G97], [S03].
- (ii) Various highly important present day computations routinely involve matrices made strongly (r, ξ) -concentrated and hence (r, ξ) -concentrated, for reasonably small integers r and small positive ξ , by means of some standard preconditioning techniques.

The next critical issue is whether we can decrease the computational cost of a CG iteration, which is reduced essentially to computing or closely approximating the product of the matrix M by a vector.

Next we prove that strongly (r, ξ) -concentrated matrices are also (r, ξ) -HSS matrices, and so, by applying our accelerated variant of FMM, we can approximate the product of such an $n \times n$ average matrix M by a vector significantly faster than by applying the known algorithms, which involve $(2n - 1)n$ flops.

Theorem 6.3. *If an $n \times n$ spd matrix M is strongly r -concentrated, then the numerical rank of any of its off-diagonal submatrix is at most r .*

Proof. Since M is a strongly r -concentrated matrix, at least $n - r$ its singular values are clustered about a certain value s . Change all the singular values by assigning to them this value s and denote the resulting matrix $\widehat{M} = M + E$ where the matrix E has numerical rank at most r and where $\widehat{M} = U\widehat{S}U^T$, for $\widehat{S} = sI_n$ and an orthogonal matrix U . Note that in this case $\widehat{M} = sI_n$, and so every off-diagonal submatrix of \widehat{M} is filled with 0s. Therefore the matrices M and E share all their off-diagonal submatrices. Consequently numerical rank of such a submatrix cannot exceed $\text{nrnk}(E) \leq r$. \square

Corollary 6.1. *If an $n \times n$ spd matrix M is strongly r -concentrated, then we can approximate the solution of a linear system of n equations $M\mathbf{x} = \mathbf{b}$ by using $O(r^2 n \log(n)) + \gamma(M)$ flops provided that one can compute generators of length at most r for a r -HSS approximate representation of the matrix M by using $\gamma(M)$ flops.*

By virtue of Theorem 6.3 the neutered block columns in the r -HSS representation of the matrix M have numerical ranks at most r . By virtue of our results of the previous subsection, $\gamma(M) = O(rn)$ in the case of the average matrix M , implying respective acceleration of the CG algorithms.

⁹Hereafter we use the acronym *spd*.

Remark 6.2. Extension to nonsymmetric inputs. Recall that any nonsingular linear system $V\mathbf{x} = \mathbf{b}$ is equivalent to the spd linear systems $V^T V\mathbf{x} = \mathbf{c}$ and $VV^T \mathbf{y} = \mathbf{b}$ for $\mathbf{c} = V^T \mathbf{b}$ and $\mathbf{x} = V^T \mathbf{y}$. Therefore we can extend our results to a nonsymmetric nonsingular linear system of equations $A\mathbf{y} = \mathbf{f}$ by means of symmetrization of a matrix A in any of the two ways, $A \rightarrow M = A^T A$ or $A \rightarrow M = AA^T$, and then application of the CG algorithms to the matrix M defined implicitly by the above products, and never computed explicitly. This leads to the CG normal equation error method and the CG normal equation error method, respectively (cf. [GL13, Section 11.3.9]), to which we can extend our study of the CG method.

Appendix: Definitions and Auxiliary Results

A General Matrices

For simplicity, in the Appendix we mostly assume dealing with real matrices, but can readily extend all our study to the complex case.

1. I_g is a $g \times g$ identity matrix. $O_{k,l}$ is the $k \times l$ matrix filled with zeros.
2. $(B_1 \mid B_2 \mid \cdots \mid B_k)$ is a block vector of length k , and $\text{diag}(B_1, B_2, \dots, B_k)$ is a $k \times k$ block diagonal matrix, in both cases with blocks B_1, B_2, \dots, B_k .
3. $W_{k,l}$ denotes the $k \times l$ leading (that is, northwestern) block of an $m \times n$ matrix W .
4. $W = S_{W,\rho} \Sigma_{W,\rho} T_{W,\rho}^T$ is compact SVD of a matrix W of rank ρ with $S_{W,\rho}$ and $T_{W,\rho}$ real orthogonal or unitary matrices of its singular vectors and $\Sigma_{W,\rho} = \text{diag}(\sigma_j(W))_{j=1}^\rho$ the diagonal matrix of its singular values in non-increasing order, $\sigma_1(W) \geq \sigma_2(W) \geq \cdots \geq \sigma_\rho(W) > 0$.
5. $W^+ = T_{W,\rho} \Sigma_{W,\rho}^{-1} S_{W,\rho}^T$ is the Moore–Penrose pseudo inverse of the matrix W .
6. $\|W\| = \sigma_1(W)$ and $\|W\|_F = (\sum_{j=1}^\rho \sigma_j^2(W))^{1/2} \leq \sqrt{n} \|W\|$ denote its spectral and Frobenius norms, respectively. ($\|W^+\| = \frac{1}{\sigma_\rho(W)}$; $\|UW\| = \|W\|$ and $\|WU\| = \|W\|$ if the matrix U is unitary.)
7. $\kappa(W) = \|W\| \|W^+\| = \sigma_1(W)/\sigma_\rho(W) \geq 1$ denotes the condition number of a matrix W . A matrix is called *ill-conditioned* if its condition number is large in context and is called *well-conditioned* if this number $\kappa(W)$ is reasonably bounded. (An $m \times n$ matrix is ill-conditioned if and only if it has a matrix of a smaller rank nearby, and it is well-conditioned if and only if it has full numerical rank $\min\{m, n\}$.)
8. The following theorem is implied by [S98, Corollary 1.4.19] for $P = -C^{-1}E$:

Theorem A.1. Suppose C and $C + E$ are two nonsingular matrices of the same size and

$$\|C^{-1}E\| = \theta < 1.$$

Then

$$\|(C + E)^{-1} - C^{-1}\| \leq \frac{\theta}{1 - \theta} \|C^{-1}\|;$$

e.g., $\|(C + E)^{-1} - C^{-1}\| \leq 0.5 \|C^{-1}\|$ if $\theta \leq 1/3$.

B Gaussian Matrices

Theorem B.1. Suppose that A is an $m \times n$ matrix of full rank $k = \min\{m, n\}$, F and H are $r \times m$ and $n \times r$ matrices, respectively, for $r \leq k$, and the entries of these two matrices are nonconstant linear combinations of finitely many i.i.d. random variables v_1, \dots, v_h .

Then the matrices F , FA , H , and AH have full rank r

(i) with probability 1 if v_1, \dots, v_h are Gaussian variables and

(ii) with a probability at least $1 - r/|\mathcal{S}|$ if they are random variables sampled under the uniform probability distribution from a finite set \mathcal{S} having cardinality $|\mathcal{S}|$.

Proof. The determinant, $\det(B)$, of any $r \times r$ block B of a matrix F , FA , H , or AH is a polynomial of degree r in the variables v_1, \dots, v_h , and so the equation $\det(B) = 0$ defines an algebraic variety of a lower dimension in the linear space of these variables (cf. [BV88, Proposition 1]). Clearly, such a variety has Lebesgue and Gaussian measures 0, both being absolutely continuous with respect to one another. This implies part (i) of the theorem. Derivation of part (ii) from a celebrated lemma of [DL78], also known from [Z79] and [S80], is a well-known pattern, specified in some detail in [PW08]. \square

Lemma B.1. (Rotational invariance of a Gaussian matrix.) *Suppose that k , m , and n are three positive integers, G is an $m \times n$ Gaussian matrix, and S and T are $k \times m$ and $n \times k$ orthogonal matrices, respectively. Then SG and GT are Gaussian matrices.*

We state the following results and estimates for real matrices, but similar estimates in the case of complex matrices can be found in [D88], [E88], [CD05], and [ES05]:

Definition B.1. Norms of random matrices and expected value of a random variable. Write $\nu_{m,n} = \|G\|$, $\nu_{m,n}^+ = \|G^+\|$, and $\nu_{m,n,F}^+ = \|G^+\|_F$, for a Gaussian $m \times n$ matrix G , and write $\mathbb{E}(v)$ for the expected value of a random variable v . ($\nu_{m,n} = \nu_{n,m}$, $\nu_{m,n}^+ = \nu_{n,m}^+$, and $\nu_{F,m,n} = \nu_{F,n,m}$, for all pairs of m and n .)

Theorem B.2. (Cf. [DS01, Theorem II.7].) *Suppose that m and n are positive integers, $h = \max\{m, n\}$, $t \geq 0$. Then*

- (i) *Probability $\{\nu_{m,n} > t + \sqrt{m} + \sqrt{n}\} \leq \exp(-t^2/2)$ and*
- (ii) *$\mathbb{E}(\nu_{m,n}) < 1 + \sqrt{m} + \sqrt{n}$.*

Theorem B.3. *Let $\Gamma(x) = \int_0^\infty \exp(-t)t^{x-1}dt$ denote the Gamma function and let $x > 0$. Then*

- (i) *Probability $\{\nu_{m,n}^+ \geq m/x^2\} < \frac{x^{m-n+1}}{\Gamma(m-n+2)}$ for $m \geq n \geq 2$,*
- (ii) *Probability $\{\nu_{n,n}^+ \geq x\} \leq 2.35\sqrt{n}/x$ for $n \geq 2$,*
- (iii) *$\mathbb{E}(\nu_{m,n}^+) \leq e\sqrt{m}/|m-n|$, provided that $m \neq n$ and $e = 2.71828\dots$*

Proof. See [CD05, Proof of Lemma 4.1] for part (i), [SST06, Theorem 3.3] for part (ii), and [HMT11, Proposition 10.2] for part (iii). \square

The probabilistic upper bounds of Theorem B.3 on $\nu_{m,n}^+$ are reasonable already in the case of square matrices, that is, where $m = n$, but become much stronger as the difference $|m - n|$ grows large.

Theorems B.2 and B.3 combined imply that an $m \times n$ Gaussian matrix is well-conditioned unless the integer $m + n$ is large or the integer $|m - n|$ is close to 0. With some grain of salt we can still consider such a matrix well-conditioned even where the integer $|m - n|$ is small or vanishes provided that the integer m is not large. Clearly, these properties can be extended immediately to all submatrices.

Acknowledgements: Our research has been supported by NSF Grant CCF 1116736 and PSC CUNY Award 68862-00 46.

References

- [A94] O. Axelsson, *Iterative Solution Methods*, Cambridge Univ. Press, Cambridge, England, 1994.
- [AMT10] H. Avron, P. Maymounkov, S. Toledo, Blendenpik: Supercharging LAPACKs least-squares solver, *SIAM Journal on Scientific Computing*, **32**, 1217–1236, 2010.
- [B10] S. Börm, *Efficient Numerical Methods for Non-local Operators: \mathcal{H}^2 -Matrix Compression, Algorithms and Analysis*, European Math. Society, 2010.
- [BGH03] S. Börm, L. Grasedyck, W. Hackbusch, Introduction to Hierarchical Matrices with Applications, *Engineering Analysis with Boundary Elements*, **27**, 5, 405–422, 2003.
- [BGP05] A. Bini, L. Gemignani, V. Y. Pan, Fast and Stable QR Eigenvalue Algorithms for Generalized Semiseparable Matrices and Secular Equation, *Numerische Mathematik*, **100**, 3, 373–408, 2005.
- [BV88] W. Bruns, U. Vetter, *Determinantal Rings, Lecture Notes in Math.*, **1327**, Springer, 1988.

- [CCF04] M. Charikar, K. Chen, M. Farach-Colton, Finding Frequent Items in Data Streams, *Theoretical Computer Science*, **312**, 1, 3–15, 2004.
- [CD05] Z. Chen, J. J. Dongarra, Condition Numbers of Gaussian Random Matrices, *SIAM. J. on Matrix Analysis and Applications*, **27**, 603–620, 2005.
- [CGR88] J. Carrier, L. Greengard, V. Rokhlin, A Fast Adaptive Algorithm for Particle Simulation, *SIAM Journal on Scientific Computing*, **9**, 669–686, 1988.
- [D88] J. Demmel, The Probability That a Numerical Analysis Problem Is Difficult, *Math. of Computation*, **50**, 449–480, 1988.
- [DE] *Dillon Engineering*, http://www.dilloneng.com/fft_ip/parallel-fft.
- [DL78] R. A. Demillo, R. J. Lipton, A Probabilistic Remark on Algebraic Program Testing, *Information Processing Letters*, **7**, 4, 193–195, 1978.
- [DS01] K. R. Davidson, S. J. Szarek, Local Operator Theory, Random Matrices, and Banach Spaces, in *Handbook on the Geometry of Banach Spaces* (W. B. Johnson and J. Lindenstrauss editors), pages 317–368, North Holland, Amsterdam, 2001.
- [E88] A. Edelman, Eigenvalues and Condition Numbers of Random Matrices, *SIAM J. on Matrix Analysis and Applications*, **9**, 4, 543–560, 1988.
- [EGH13] Y. Eidelman, I. Gohberg, I. Haimovici, *Separable Type Representations of Matrices and Fast Algorithms*, volumes 1 and 2, Birkhäuser, 2013.
- [ES05] A. Edelman, B. D. Sutton, Tails of Condition Number Distributions, *SIAM J. on Matrix Analysis and Applications*, **27**, 2, 547–560, 2005.
- [F77] R. Freivalds, Probabilistic Machines Can Use Less Running Time, *IFIP Congress 1977*, 839842, 1977.
- [G97] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.
- [GH03] L. Grasedyck, W. Hackbusch, Construction and Arithmetics of H-Matrices, *Computing*, **70**, 4, 295–334, 2003.
- [GL13] G. H. Golub, C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland, 2013 (fourth edition).
- [GR87] L. Greengard, V. Rokhlin, A Fast Algorithm for Particle Simulation, *Journal of Computational Physics*, **73**, 325–348, 1987.
- [H02] N. J. Higham, *Accuracy and Stability in Numerical Analysis*, SIAM, Philadelphia, 2002.
- [HMST11] N. Halko, P. G. Martinsson, Y. Shkolnisky, M. Tygert, An Algorithm for the Principal Component Analysis of Large Data Sets, *SIAM J. Scientific Computation*, **33**, 5, 2580–2594, 2011.
- [HMT11] N. Halko, P. G. Martinsson, J. A. Tropp, Finding Structure with Randomness: Probabilistic Algorithms for Approximate Matrix Decompositions, *SIAM Review*, **53**, 2, 217–288, 2011.
- [M11] M. W. Mahoney, Randomized Algorithms for Matrices and Data, *Foundations and Trends in Machine Learning*, NOW Publishers, **3**, 2, 2011.
- [MRT05] P. G. Martinsson, V. Rokhlin, M. Tygert, A Fast Algorithm for the Inversion of General Toeplitz Matrices, *Comput. Math. Appl.*, **50**, 741–752, 2005.
- [P01] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York, 2001.

- [P15] V. Y. Pan, Transformations of Matrix Structures Work Again, *Linear Algebra and Its Applications*, **465**, 1–32, 2015.
- [PQY15] V. Y. Pan, G. Qian, X. Yan, Random Multipliers Numerically Stabilize Gaussian and Block Gaussian Elimination: Proofs and an Extension to Low-rank Approximation, *Linear Algebra and Its Applications*, **481**, 202–234, 2015.
- [PSZ15] V. Y. Pan, J. Svadlenka, L. Zhao, Estimating the Norms of Circulant and Toeplitz Random Matrices and Their Inverses, *Linear Algebra and Its Applications*, **468**, 197–210, 2015.
- [PW08] V. Y. Pan, X. Wang, Degeneration of Integer Matrices Modulo an Integer, *Linear Algebra and Its Applications*, **429**, 2113–2130, 2008.
- [PZa] V. Y. Pan, L. Zhao, Numerically Safe Gaussian Elimination with No Pivoting, arxiv 1501.05385 CS, submitted on January 22, 2015, revised in June 2016.
- [RST09] V. Rokhlin, A. Szlam, M. Tygert, A Randomized Algorithm for Principal Component Analysis, *SIAM Journal on Matrix Analysis and Applications*, **31**, **3**, 1100–1124, 2009.
- [RT08] V. Rokhlin, M. Tygert, A Fast Randomized Algorithm for Overdetermined Linear Least-squares Regression, *Proc. Natl. Acad. Sci. USA*, **105**, **36**, 13212–13217, 2008.
- [S80] J. T. Schwartz, Fast Probabilistic Algorithms for Verification of Polynomial Identities, *Journal of ACM*, **27**, **4**, 701–717, 1980.
- [S98] G. W. Stewart, *Matrix Algorithms, Vol I: Basic Decompositions*, SIAM, Philadelphia, 1998.
- [S03] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003.
- [S06] T. Sarlós, Improved Approximation Algorithms for Large Matrices via Random Projections. *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, 143–152, 2006.
- [SST06] A. Sankar, D. Spielman, S.-H. Teng, Smoothed Analysis of the Condition Numbers and Growth Factors of Matrices, *SIAM J. on Matrix Analysis and Applcs.*, **28**, **2**, 446–476, 2006.
- [T00] E.E. Tyrtyshnikov, Incomplete Cross-Approximation in the Mosaic-Skeleton Method, *Computing*, **64**, 367–380, 2000.
- [T11] J. A. Tropp, Improved Analysis of the Subsampled Randomized Hadamard Transform, *Adv. Adapt. Data Anal.*, **3**, **1–2** (Special issue "Sparse Representation of Data and Images"), 115–126, 2011. Also arXiv math.NA 1011.1595.
- [TZ12] M. Thorup, Y. Zhang, Tabulation-based 5-independent Hashing with Applications to Linear Probing and Second Moment Estimation, *SIAM J. on Computing*, **41**, **2**, 293–331, 2012.
- [VVG05] R. Vandebril, M. Van Barel, G. Golub, N. Mastronardi, A Bibliography on Semiseparable Matrices, *Calcolo*, **42**, **3–4**, 249–270, 2005.
- [VVM07/08] R. Vandebril, M. Van Barel, N. Mastronardi, *Matrix Computations and Semiseparable Matrices* (Volumes 1 and 2), The Johns Hopkins University Press, Baltimore, Maryland, 2007.
- [VVVF10] M. Van Barel, R. Vandebril, P. Van Dooren, K. Frederix, Implicit Double Shift *QR*-algorithm for Companion Matrices, *Numerische Mathematik*, **116**, 177–212, 2010.
- [W14] D.P. Woodruff, Sketching as a Tool for Numerical Linear Algebra, *Foundations and Trends in Theoretical Computer Science*, **10**, **1–2**, 1–157, 2014.
- [X12] J. Xia, On the Complexity of Some Hierarchical Structured Matrix Algorithms, *SIAM J. Matrix Anal. Appl.*, **33**, 388–410, 2012.
- [X13] J. Xia, Randomized Sparse Direct Solvers, *SIAM J. Matrix Anal. Appl.*, **34**, 197–227, 2013.

- [XXCB14] J. Xia, Y. Xi, S. Cauley, V. Balakrishnan, Superfast and Stable Structured Solvers for Toeplitz Least Squares via Randomized Sampling, *SIAM J. Matrix Anal. Appl.*, **35**, 44–72, 2014.
- [XXG12] J. Xia, Y. Xi, M. Gu, A Superfast Structured Solver for Toeplitz Linear Systems via Randomized Sampling, *SIAM J. Matrix Anal. Appl.*, **33**, 837–858, 2012.
- [Z79] R. E. Zippel, Probabilistic Algorithms for Sparse Polynomials, *Proceedings of EUROSAM'79, Lecture Notes in Computer Science*, **72**, 216–226, Springer, Berlin, 1979.